504-REN-96/003
CSC10031818

MISSION OPERATIONS AND DATA SYSTEMS DIRECTORATE

Renaissance Generic Architecture

Version 2.0

Volume 1 of 2

May 1996

National Aeronautics and Space Administration

Goddard Space Flight Center
Greenbelt, Maryland

Renaissance Generic Architecture

Version 2.0

Volume 1 0f 2

May 1996

Prepared Under Contract NAS5-31000
Task Assignment 05-601

Approved By:

Gary F. Meyers Date
Systems Engineering Office, Code 504

Goddard Space Flight Center
Greenbelt, Maryland

Preface

This report describes the architecture prepared by the Renaissance Team for the second generation aproach for Mission Operations and Data Systems Directorate (MO&DSD)-developed ground data systems. This second generation architecture is predicated on a willingness to evolve more rapidly toward more effective, less costly implementations of mission ground data systems. This willingness provides the impetus for a design approach freed from dependency on MO&DSD legacy. The goal for this effort was definition of a design that could be implemented for missions launching as soon as 1997 to 1998, achieving significant advances over the MO&DSD legacy design developed as Renaissance first generation.

This design effort started during Fiscal Year (FY) 1995 and was completed in FY96, under the auspices of the Renaissance program in Code 504.

This document is under the configuration management of the Systems Engineering Office.

Configuration Change Requests (CCRs) to this document shall be submitted to the Systems Engineering Office, along with supportive material justifying the proposed change. Changes to this document shall be made by document change notice (DCN) or by complete revision.

Questions and proposed changes concerning this document shall be addressed to:

Gary F. Meyers, Systems Engineering Office
Code 504
Goddard Space Flight Center
Greenbelt, Maryland 20771

Abstract

This document contains the Renaissance Generic Architecture, Version 2, developed by the Renaissance program of Mission Operations and Data Systems Directorate (MO&DSD) for future mission ground data system development. It provides a framework for mission system development based on integration of off-the-shelf software and hardware. Objectives are major reductions in development time and cost, operations cost, and improved flexibility to meet future mission needs more effectively, including insertion of new technologies.

The architecture design includes elements of software, hardware, data and communications, and addresses general operations concepts expected for future missions. The design supports mission systems ranging from single, dedicated local area networks including ground station equipment, to extended multi-site systems.

The design is based on a hierarchy of networked processes, with standard interfaces between the layers of the hierarchy. The hierarchy is conceptually based on physical network domains, i.e., wide area networks (WANs), local area networks (LANs), and smaller components. The design allows future spacecraft to be integrated in as another node or LAN in the mission system. External and internal interfaces are based on Internet standards, with supplementary material where appropriate.

Keywords: *Renaissance, architecture, ground data system, spacecraft data processing.*

Contents

Volume 1:

Section 1. Introduction

Section 2. System Architecture

Section 3. Software Architecture

Section 4. Application Design

Section 5. Hardware Architecture

Abbreviations and Acronyms

Glossary

Bibliography

List of Figures

List of Tables

Volume 2:

Appendix A. Second Generation Concepts

Appendix B. Second Generation Mission Requirements

Appendix C. Architecture Alternatives and Analysis

Appendix D. Capabilities and Design Drivers

Appendix E. University Mission Design

Appendix F. ASIST Oriented Design

Appendix G. NASA-Integrated Mission Design

Appendix H. Technical Performance Approaches

Section 1. Introduction

1.1 Architecture Purpose

This document describes the second release of the Renaissance proposed Ground Data System (GDS) architecture for supporting future unmanned scientific space missions. The missions to be considered for second generation implementation are those whose launch date is 1998 or later, i.e., those for whom the implementation of the spacecraft and GDS will not begin until mid to late FY96. The second release defines an architecture that provides a foundation for improved mission development and operations effectiveness, can be initially implemented in the immediate future, can evolve with anticipated changes in mission operations and implementations technology, and is not constrained by current approaches.

The objectives of this design include a major reduction in the development time and cost, efficient operations, evolution with changes expected in spacecraft implementation, and support for operations by non-Goddard Space Flight Center (GSFC) organizations. A basic design objective for the second generation was to maximize the ability to use available components and minimize the need for development of components. The intent of the design effort was to create designs friendly for use not only of GSFC-developed components, but also those from other government sources and commercial off-the-shelf (COTS) vendors. By increasing the available components, the ability to meet varying mission needs without development is significantly improved from the legacy systems at GSFC.

The architecture incorporates a generic architecture framework and system-level design of the components and connections for the entire GDS. The architecture provides a set of options in specific mission system design and implementation, along with assessment of those factors that will aid in selection of the final design. The document includes sample mission designs using off-the-shelf components based on the higher level design. This document provides design guidance for selecting from available products and implementing systems based on those products.

1.2 Architecture Development Process

The process began with concurrent activities to assess expected technology changes, to develop a concept of future operational needs and ground data system requirements, and an initial study to identify and analyze alternative design approaches for ground data systems. The Emerging Technology and Architecture action team reports are the results of the first and last of these activities. The operations concepts and mission requirements captured in Appendices A and B, show the results of these activities.

Following these, two additional activities were performed: an assessment of existing applicable COTS products, and a development of alternative architecture frameworks for system design. The COTS assessment task met with about a half-dozen vendors and contacted others for

information. Based on these meetings, a preliminary assessment of the functions covered and the implementation impacts of these products was prepared and internally documented. The framework analysis task identified and evaluated four architecture frameworks. A summary of the evaluation of these frameworks and the rationale for the selected framework are presented in Appendix C.

The following final design activity addressed the varying alternatives, and developed the final architecture presented in this document. Several strawman designs based on the architecture were developed and are presented in Appendices E - G. Finally, a preliminary approach to addressing technical performance in the architecture was prepared and is included as Appendix H.

1.3 Architecture Scope

The functionality and capability of the systems to be considered within the scope of the second generation architecture are expanded from that of MO&DSD legacy mission systems. The system scope has been expanded to include added functions invoked at the ground stations, the spacecraft, and the customer payload analysis centers. Capabilities have also been included to support mission integration and test, and mission maintenance activities. The intent is to provide as global a coverage of mission support capabilities as is feasible.

The most extreme change is the inclusion of spacecraft functionality to address functions, e.g., monitoring spacecraft health and safety, that may in the future be moved on-board the spacecraft and to design these so that such movement is readily incorporated.

At the payload analysis center, infrastructure functionality to integrate the science center and to integrate mission planning and accounting is the primary area included. Generation of standard products is included. None of the true scientific or payload-unique analysis functions are included.

Mission integration and test capabilities require added flexibility in configuring systems so that very simple and inexpensive solutions are available to address low-level testing. Development and maintenance are connected elements incorporating the same basic infrastructure but with different applications. This design has not addressed the unique applications but provides an integration approach.

1.4 The Architecture

The resulting architecture is different from the typical result of MO&DSD architecture development. It is a domain architecture and, therefore, generalized to support multiple space support domain environments. It also is intended to support wide use of existing application and infrastructure products for mission implementation, and to allow ready evolution in the future. The architecture process led to the understanding that there were some important issues about application interfaces, including:

- Widely divergent boundaries and capabilities for applications with essentially common functions

- Widely divergent component types: packages, objects, agents, etc.
- A gross lack of agreement on operations concepts and the best way to provide the needed capability
- A lack of standards for high-level interfaces that have significant commercial support

The result is an architecture with components less specified than is typically expected. Instead of hard component specifications, one gets interfaces and guidelines for instantiation of the architecture. The interfaces and allocations of capability are developed around a layered model of services loosely based on the Open System Interconnect (OSI) reference model. The interfaces between layers are defined around established standards to support use of existing applications to provide a stable interface. The critical element of this distributed architecture is the "network transparency" layer, which provides the services and Application Program Interfaces (APIs) to seemlessly connect applications across the entire system network. The architecture is object neutral, in that both object-oriented and conventional designs are allowed. A specific framework, e.g., Taligent, is not selected because of the limitations this would impose on use of off-the-shelf applications.

These features are what contrast this architecture with the first generation developed for the Advanced Composition Explorer (ACE). Specific applications and interfaces are excluded to allow variations both in current implementations and for future development. The interfaces are APIs instead, with the specific message exchange not uniquely defined. No specific framework for the infrastructure is defined either. Both the number and the type of components are allowed to change. The essential, fixed elements are the locations and applicable standards of the boundaries between service layers.

There are two significant advantages: no products are formally ruled out because they don't fit the architecture, and the instantiation can be chosen to take advantage of existing product suites with well-defined boundaries and interfaces. This approach then minimizes custom development by allowing most effective use of off-the-shelf products.

The applications are expected to correspond generally to the three-tier client/server model. This model consists of an operator interface client, and intermediate, and possibly compound, application, and an information server. Simpler application forms are not excluded, but the domain functions appear generally to force the more complex design to reach the needed flexibility in system configuration.

The label "generic" architecture has been applied to this result of a generalized approach. The next level of specification then is the instantiation level, which can be developed directly from the generic definitions.

1.5 Document Structure

The full document is divided into two volumes: the first contains the definition of the architecture, and the second contains support information in the form of appendices. Section 2 of this volume is the overview presentation of the features of the system architecture. Sections 3

and 4 detail the software architecture for the support services and the applications, respectively. Section 5 summarizes the approach for hardware for the architecture.

Volume 2 contains eight appendices relating information extending the detail and providing information on the trades and analysis leading to the definition of the selected architecture. Appendices A and B provide a discussion of the operations concepts and requirements profiled for this generation of mission systems. Appendices C and D describe the preliminary analyses that led to the selection of the architecture features. Finally, Appendices E, F and G contain representative designs for future missions, based on the architecture. Appendix H provides an analysis of the application of the architecture features to meeting mission technical performance needs.

Section 2. System Architecture

The generic Renaissance architecture provides the basis for implementing mission systems for near and far future missions to be developed and operated with National Aeronautical and Space Administration (NASA) support. It supports the expected radical changes in that support and in the relationship between NASA and its mission customers. This section provides a brief summary of the requirements supported by the architecture and a presentation of the architecture.

The architecture has been devised around a small set of conceptual features:

- Scalable to the wide range of mission needs
- Flexible enough to support large variations in functional allocation among ground and space components
- Adaptable to operations character changes from hands-on to fully automated
- Stable provision of mechanisms for interprocess data exchange and control
- Implementable with a wide range of off-the-shelf application and infrastructure components

These characteristics drive the approach of a multilayer architecture, using the layering to enhance flexibility and standard interprocess mechanisms to enforce a common interface. The summary of requirements and context is provided in Section 2.1. The system architecture features are presented and explained in Section 2.2. Section 2.3 discusses the approach to managing the variations over a mission life of the components and interconnections. Section 2.4 explains the approach to external interfaces, and Section 2.5 presents an introduction to mission systems instantiation based on the generic architecture.

2.1 System Requirements Summary

2.1.1 Mission System Scope and Context

The mission system is inclusive of all elements directly associated with the development and operations of the mission. This includes any spacecraft, the operational ground data system of spacecraft bus and payload management, payload data production, spacecraft and terrestrial subsystem development and test capabilities, and any communications elements needed to integrate these. Excluded are entities not directly connected with the mission.

Figure 2-1 is a context diagram with the external entities identified as generic categories. Each external entity is only representative, and may be included in some mission systems. Examples could include the spacecraft development and Integration and Test (I&T) facilities, some of the NASA institutional elements, and customer entities. The key is whether or not they are established as a part of the mission system in architecture and operations.

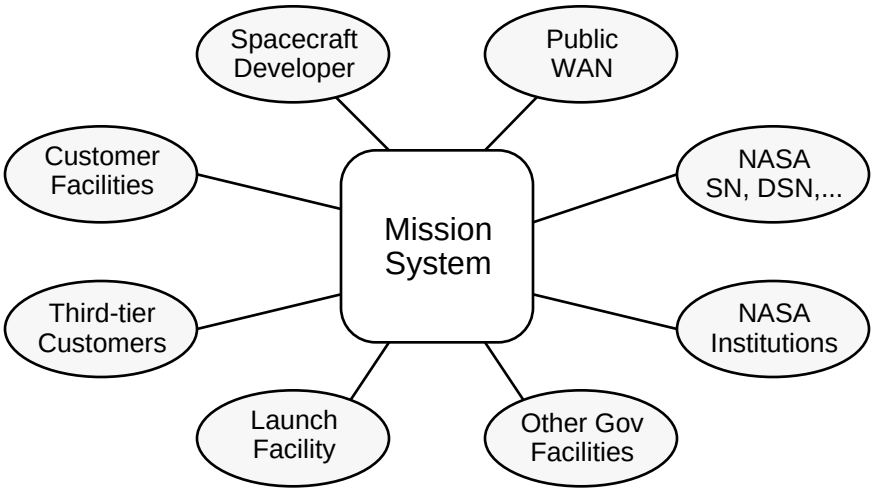

Figure 2-1. Mission System Context

2.1.2 Mission System Programmatic Requirements

Many primary architecture drivers are programmatic requirements. These force many of the design decisions made in providing the capabilities discussed in section 2.1.1. The following represent just those with the greatest impact; a more thorough discussion is presented in Appendix D.

- Reduce cost of implementation and operation: MO&DSD must be able to deliver systems that cost less to build and less to operate. This will enable the addition of more advanced features within flat or even decreasing budgets for NASA and research. Target costs are for an order of magnitude reduction for implementation. A derivative of the cost needs is use of off-the-shelf components for most of mission implementation.
- Support customers better by providing a more flexible response: Eliminate a dependency on traditional institutional approaches and address customer desires for more direct access to the mission system.
- Simplify operations: Provide mechanisms to simplify operations. This addresses the needs for operations cost reductions and the desire for customer-based operation. Many changes in available components, both software and hardware are anticipated, e.g., spacecraft autonomy capabilities and the Global Positioning System (GPS). Operations must evolve to decrease mission costs and to provide better support for the end deliverables and customers. Many of today's manual processes will be replaced with automated systems, and more mission control will be placed in the hands of the customers.
- Support major changes in system implementation: Concepts for extremely different missions, e.g., the New Millennium, must be included in the mix of mission types to be

supported. This type of change is occurring rapidly, linked to accelerated mission life cycles, and new technologies. Low cost is driving a rapid technology change.

- Feasibility of immediate implementation: It is necessary to enable use of more flexible and cost-effective systems for missions currently entering the implementation phase, but it is not possible to develop a complete new system with the time and money available. The system must enable use of current off-the-shelf products from anywhere, commercial or government, to achieve rapid improvement without sacrificing future gains.
- Flexibility in NASA support: NASA is expected to continue to offer a wide variety of institutional support services. While these have frequently been the entire operational context for past missions, it is expected that future missions will have the freedom to select those services desired for the individual mission. In some cases, the mission programs, e.g., MIDEX, will provide for two or more contexts from which to select. The anticipated services are extensions of current and planned NASA activities.

2.2 System Architecture Views

The system architecture is configured around a set of relationships between components and their services. The components and relationships are shown in this section in three views: the system component interconnection view, the system services domain view, and the layered system view. Each view is presented in the following three sections.

2.2.1 System Component Interconnection View

The system component interconnection view is the most intuitive view of the system architecture because it is closely related to the most physically apparent components. This view defines a hierarchy of the components of the system and the connections between them. Figure 2-2 shows this abstract representation of the generic components of the architecture, from spacecraft through end customer payload data analysis.

Because this is a generic architecture to be applicable to many systems, the components are generic, that is, the types of components are specified but the allocation of capabilities to these components is not uniquely specified. The ability to change the number of each type of component and the allocation of capabilities to each specific component is one of the major aspects of flexibility built into this architecture.

2.2.1.1 Global Level

At the broadest level, shown at the bottom of the figure, the components are logical subsystems, which in most cases are closely aligned with physical entities: the facilities provided by NASA, the customer, and third-party contractors to perform mission activities. These are linked by communications facilities for terrestrial and space communications and for interfacing between the two. The spacecraft is shown as a logically equivalent subsystem attached to the common network. This figure does not indicate the time dependence of connections during development, and following launch.

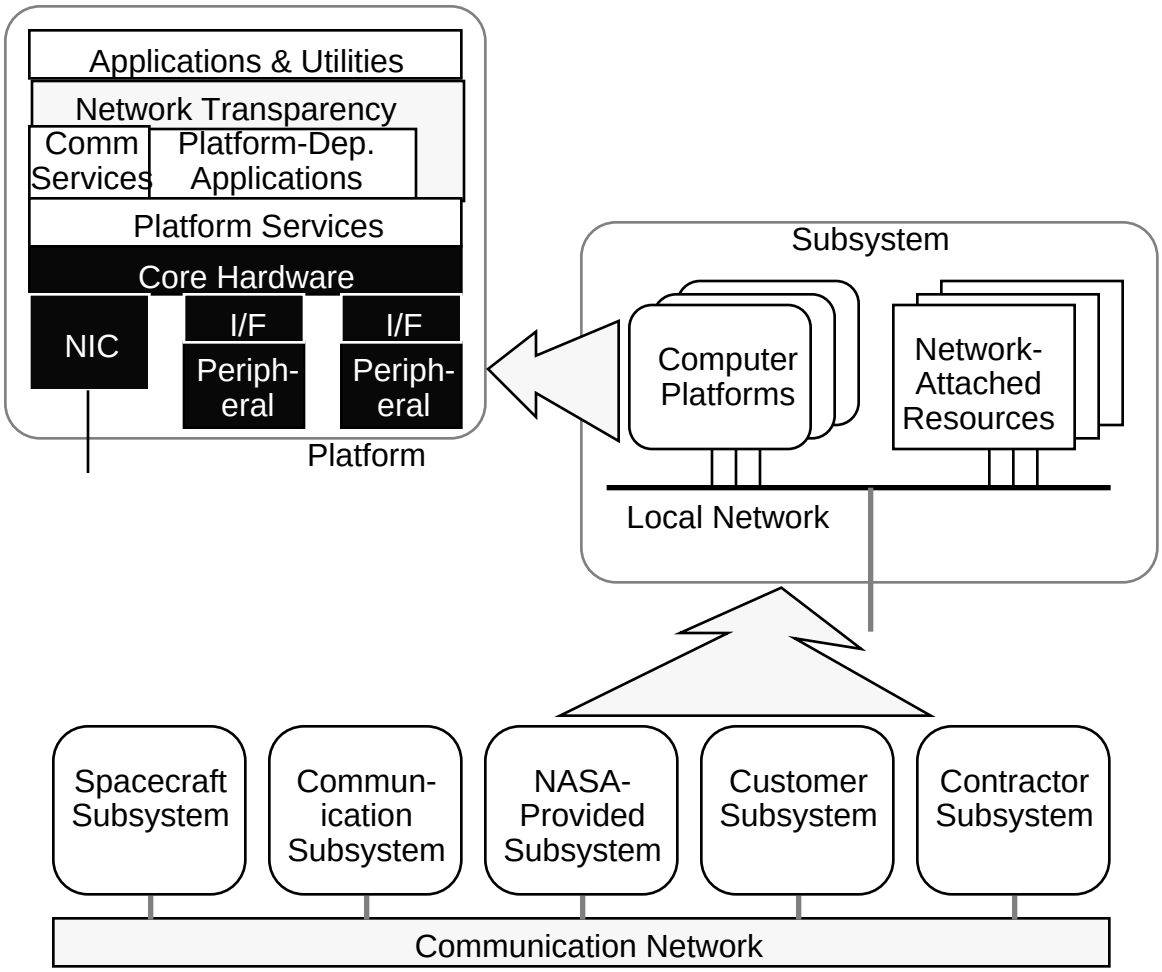

Figure 2-2. System Components Interconnection View

The communications segment may be partitioned among more than a single source. Typically, both public and private resources are used for terrestrial communications. Space communications may integrate private resources, and publicly owned ones from NASA, from other U.S. government agencies, and from both commercial and international sources. These may involve different implementations for existing, shared resources and differing interfaces. Similarly, the facilities operated by NASA, the customer, and third-party contractors may incorporate shared resources that are not common to all.

In current mission systems, most subsystems are provided by NASA institutional facilities, with customer-provided science analysis facilities and contractor facilities for spacecraft development and integration and test support. Future mission systems are expected to change this. Both the MIDEX and the SMEX-Lite program activities assume a much lower dependence on NASA facility involvement, with a more concentrated facility provided by the customer containing most system capabilities. Also, current spacecraft are not truly connected to a common network, and it is only future missions, implementing new systems with protocols such as the Space

Communications Protocol Standards (SCPS) or the Internet Protocol (IP) over Consultitative Committee on Spacecraft Data Systems (CCSDS) protocols that will be connected to support common communications capabilities. Section 2.5 illustrates a few of these possibilities.

2.2.1.2 Local Area Level

At the next level of the hierarchy, as illustrated in the upper right corner of Figure 2-2, the subsystems are each composed of one or more computer platforms and network connected resources, interconnected by a network localized to the subsystem, and with a standard interface to the common system communication network. This local network provides an efficient, closely coupled set of resources for use by a mission operational entity.

The local communications network may be a simple LAN, a switched network, or a multisegment network with backbone and multiple LANs. The implementation technology and the overall topology are configured to meet the technical performance needs of the subsystem.

The subsystem computer platforms need not be homogeneous. Both UNIX and Windows NT platforms are fully supported, and limited use of platforms with other operating systems is also allowed. Network-attached resources are special purpose components that do not provide general computing capability. Examples of these include data storage subsystems, network attached printers, and specialized components such as radio frequency (RF) antenna control subsystems.

Examples of operational entities include a ground station, a test process group, and the spacecraft itself. It will not necessarily include the human interface platforms for the capabilities. Implementations vary to support variations in cost and performance based on the operational requirements and on local constraints such as environmental constraints and administrative issues. The operational unit of a local area network with applications and platforms connected is a basic building block of this architecture and is referred to as a *LAN Cell* in subsequent sections.

2.2.1.3 Platform Level

As shown in the upper left corner of Figure 2-2, the general computer platform provides a standard architecture integrating the system applications, most standard infrastructure services, and many of the hardware resources needed by these software components. The platform connects to the local network through the Network Interface Card (NIC), which provides the essential hardware and data gateway between platform-specific information representations and network representations. The platform hardware also includes the core components, i.e., Central Processing Unit (CPU), memory and data bus, and supports a variety of peripheral resources, such as Human-Computer Interface (HCI) devices, data storage devices and other specialized components, which provide local capabilities for the system. The software components include the operating system, platform-specific services such as file systems and Input-Output (IO) services, applications, network transparency services and communication services. Communications services provide the software link to interplatform communications. Network transparency services provide the critical capability for applications to interconnect, and to connect to system resources, without detailed knowledge of the system configuration and the communications implementation. Applications are all those components that provide the core

mission functionality, outside the general purpose capabilities provided by the infrastructure, and are described in greater detail in the Software Architecture and Applications Design sections.

Computer platforms and operating systems are based on use of two standard technologies: workstations and servers using UNIX with real-time extensions; and PCs, workstations and servers using Microsoft Windows NT. The former environment has provided much of the impetus for open systems implementations, and many of the available implementations for space GDS. The latter environment provides a less expensive environment and is becoming a common base for GDS-related applications. Current trends are toward common provision of many COTS applications in both environments. Use of these environments supports user interfaces in both X Windows/Motif and Windows, with cross-platform capabilities already in existence.

Network-attached resources are not further illustrated because they are widely varying special-purpose components and because they are usually acquired as integrated components at the subsystem level. There can be exceptions to this. Such resources may include embedded software but are not for general purpose computing, but to meet specific, narrowly defined needs. It is possible to consider some of the current, simpler spacecraft as network attached resources rather than subsystems, given the design of these spacecraft.

2.2.2 Layered Architecture View

The layered architecture view provides a description of the system as a set of layered services components, with the layering providing a major part of the flexibility built into the architecture. The layers have been designed around a combination of the OSI reference model and the National Institute for Standards and Technology (NIST) open systems environment (OSE) model. This combination provides the clearest approach to providing the network transparency for applications needed for our distributed approach.

Figure 2-3 illustrates the main features of this architecture approach. The figure shows a generalized computer platform over a communication network. This is an abstract representation of the elementary components of the system as shown in Figure 2-2, with all higher level components being constructed of these elements. The platform and network representation are very similar to the NIST (and DoD) technical reference model. The primary difference between this model and the NIST model is the addition of the Network Transparency layer below the applications. The equivalent OSI reference layers are shown as parenthetical labels along the left side of the platform and network components.

The Network Transparency layer is the key to this view. While this layer exists in the NIST model, here it serves as the primary interface to applications, and it has specific functions added to provide true network transparency for application access to system resources. While applications that directly access platform services and operating system interfaces are allowed, these are only for special purposes, at least some of which support network transparency, and some that NIST considers platform services, e.g., a database management system (DBMS). By setting up the network transparency interface, applications need not know where in the network the resources or other applications exist. This provides features to support movement of applications to other locations as operational changes, whether statically for long-term

performance tuning or dynamically as part of a fault recovery. It also greatly simplifies the coding of applications and the integration of existing off-the-shelf applications.

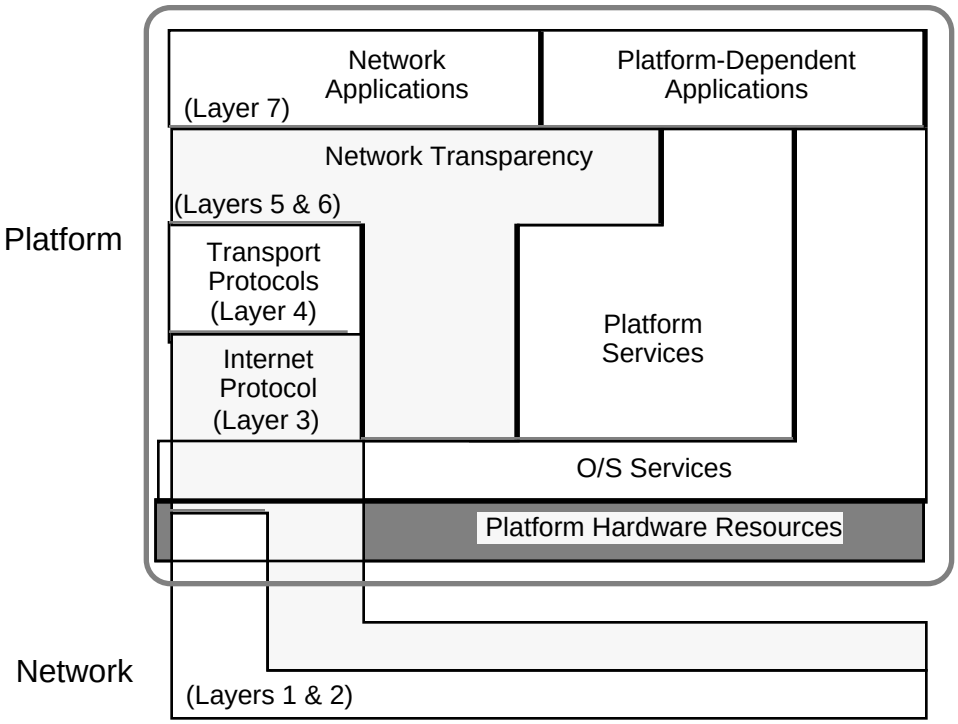

Figure 2-3. Layered Architecture View

In the following discussion, the allocation of capability to the architecture reference layers is described, where the reference layers are linked directly to the OSI reference model. For each layer, the allocated capabilities and the implied components are described.

2.2.2.1 Physical and Datalink Layer (OSI Levels 1 and 2)

The physical hardware and datalink protocol layer provide the elemental capability to transmit data packets from one node on a physical network to another (or to all on the network). It includes the physical elements, the electronic or optical elements of the network and the protocol for addressing data packets and transmitting them. The physical media allowed include anything: twisted pair or coaxial cable, optical fiber and RF transmission. The protocol implementations anticipated include Ethernet, Fiber Distributed Data Interface (FDDI), Fractional T3, Synchronous Optical Network (SONET) and Asynchronous Transfer Mode (ATM). Others are not excluded, but are limited to special cases because of compatibility.

Both wide area communications and local area communications are included in this level. Together, the composite network constitutes the essential system communications network. Some parts may not be owned by the system owners. In particular, wide area communications are more typically purchased from service providers, commercial or government, than built for

unique systems. Thus, elements of the overall network may be shared with other systems, and some security mechanisms must be included to ensure separation of service at the needed level.

Wide Area Interconnection Services provide basic digital network interconnection for system sites supporting voice, video and data services. The system needs to establish the required core communication services. These can include:

1. Basic Network Service: Data-only services, with no fixed bandwidth allocation, throughput subject to competing demand.
2. Integrated Services Data Network (ISDN): Basic and Bandwidth Allocation Control Protocol (BACP)
3. Reserved Bandwidth: Point-to-Point bandwidth reservation, scheduled directly with the network services provider, and can include voice, video and/or data.
4. Resource Reservation Protocol (RSVP): RSVP bandwidth reservation up to maximum allocated to specific customer and can include voice, video and/or data.

Customers may acquire a combination of these services from a services provider in constructing a virtual network. Owned segments are constructed from commercially available components, with few exceptions. The space segments comprise the least common elements of the network and may require more use of semicustom components.

It is anticipated that, at this level, different network segments will be constructed with different technologies and supporting different technical performance levels. This is an important factor in meeting the system needs without the need for custom construction. Both bandwidth and network reliability can be set primarily through selection of the network topology and the implementation technology.

2.2.2.2 Network Layer (OSI Level 3)

The supported network services are those provided by IP. Specification of this protocol is controlled by the Internet Engineering Task Force (IETF). The current version is v4. IPv6 is undergoing the evaluation and approval process. IPv6 includes extended addresses and security features, and such support will be needed in the future. IPv6 is currently a Draft Request for Comment (RFC) of the IETF. Supported services currently include:

1. Datagram delivery between single source and destination end system.
2. Fault tolerant WAN network due to the mesh network architecture.
3. Router-based services:
    a. Prioritized service based on source and destination addresses
    b. Service filtering based on source and destination addresses (security feature)
    c. Multicast datagram delivery

2.2.2.3 Transport Layer (OSI Level 4)

Transport level services provide the essential end-to-end transport of information across the network. They provide various quality-of-service options and support user access to the network layer for transport. These include:

1. User Datagram Protocol (UDP): unverified transport protocol, includes a multicast router capability. Accepted packets are error free based on an internal packet algorithm.

2. Transmission Control Protocol (TCP): verified, reliable host-to-host transport. This service has performance limitations over a wide area because of the latency associated with the receipt acknowledgment and retransmission requirements. Revised versions of this protocol that will address these performance problems exist in draft form.

3. Reliable Multicast Protocol (RMP): a draft standard protocol that supports verified multicast transport.

2.2.2.4 Network Transparency Layer (OSI Levels 5 and 6)

Distribution of processing and data is enabled through network transparency services. This isolates applications from the details of the operating infrastructure and enables the efficient provision of reliability and availability for processing and data access. These services provide the functionality to convert data from an application level to the transport level and hide the complexities of the network protocol stack. In general terms, the critical functionality supported includes application interconnection, security, directory services, and system management. This layer sets the primary APIs, which are independent of the platform on which the application resides.

The services identified here are general and form a complete set. Individual system instantiations may not implement all services. Also, not all services are well supported today with COTS, which may constrain support. The general services include:

1. Application interconnection: This service provides the network equivalent of the "call" service. Depending on the needs and the options available, this can include both synchronous and asynchronous interconnections and establish connections on a datagram, or a session, basis.

2. Security: This service provides the application access to the many security functions, including authentication, authorization, audit creation and nonrepudiation. The intent is that security is established on a system-wide, integrated basis. Key features include single sign-on, information privacy, and protection of system resources from unauthorized interference and destruction.

3. Directory services: These provide the mechanisms for registering names, properties and available system capabilities. This normally is a dynamic register that can be accessed to establish initial connections to accomplish an activity, reconnecting to recover from a fault, and load distribution by selecting on resource from a set of equivalent components. This is an essential information source that supports many other services.

4. System management: This service provides integrated and automated system management, including network management. The services include change management, licensing of use, creation, replication, deletion and placement of components within the system. Inherent in this is conformance to and support for configuration management.

5. Information access: This provides a network transparent access to information on the system, including data transport services, nonvolatile storage, querying for information, establishing and performing reliable transactions, and supporting concurrency control.

6. Time service: This service provides a consistent network wide access to current time in the desired units, nominally Coordinated Universal Time (UTC). Accuracy is determined by the implementation of the service.

Current commercial implementations of these services are many. Most are partial implementations, requiring combination of many of these to meet system needs. Available implementations include:

1. Domain Name System: federated node name to IP address translation.

2. X.500 Directory Services

3. RPCs, sockets and transport level interfaces (TLIs) of various styles

4. Distributed Computing Environment (DCE): includes security services, RPCs and pipes, distributed files, time and comprehensive directory services

5. Common Object Request Broker Architecture (CORBA): includes object request brokers and many object services. This design is the most complete integration of services to date.

6. Open Database Connectivity (ODBC): network connectivity to databases

7. Proprietary data services: NFS, EDA/SQL, Oracle SQLnet, Isis, Talarian RTworks, Teknekron Enterprise Messaging Service, et al.

Implementation of these services requires integration with support applications. These applications belong in the next layer, but generally belong to the class of application that maps into NIST platform support services and has access to platform specific interfaces as well as the network APIs. Section 2.2.2.5 contains more discussion.

2.2.2.5 Application Layer (OSI Level 7)

This layer includes all the applications that provide specific system capabilities. These are the capabilities that make the system unique, i.e., all the space system functionality. For this architecture these applications can, as shown in Figure 2-3, be separated into two classes: those using only network transparency services, and those accessing platform-specific services. Most system applications are intended to belong to the first class. The second class is intended to apply primarily to the support applications that need access to platform resources or are using these interfaces to achieve needed technical performance.

Applications are expected to be constructed from multiple components, distributed over the network, and interconnected through the network transparency layer. Generally, a form of three-tier client-server design is anticipated. However, current applications that do not map to this model can make forced compliance with use of a three-tier model very costly. Thus, there is no absolute restriction to this model. Additionally, there is concern that this model will not always be the most desirable and that the generic architecture should not overly constrain the implementation. This is discussed further in the software architecture section.

2.2.3 System Services Domain View

Figure 2-4 shows the essential features of the System Service Domain View. This conceptual view illustrates the mapping of system services to global and localized domains. The domains have been defined to be: global, or system wide; local area, usually mapped to a subsystem in the interconnection view, but may be mapped to a different operational entity rather than physical; and platform, only accessible within a given computer platform. The services have been organized in the diagram so those corresponding services in the different domains are aligned in vertical columns.

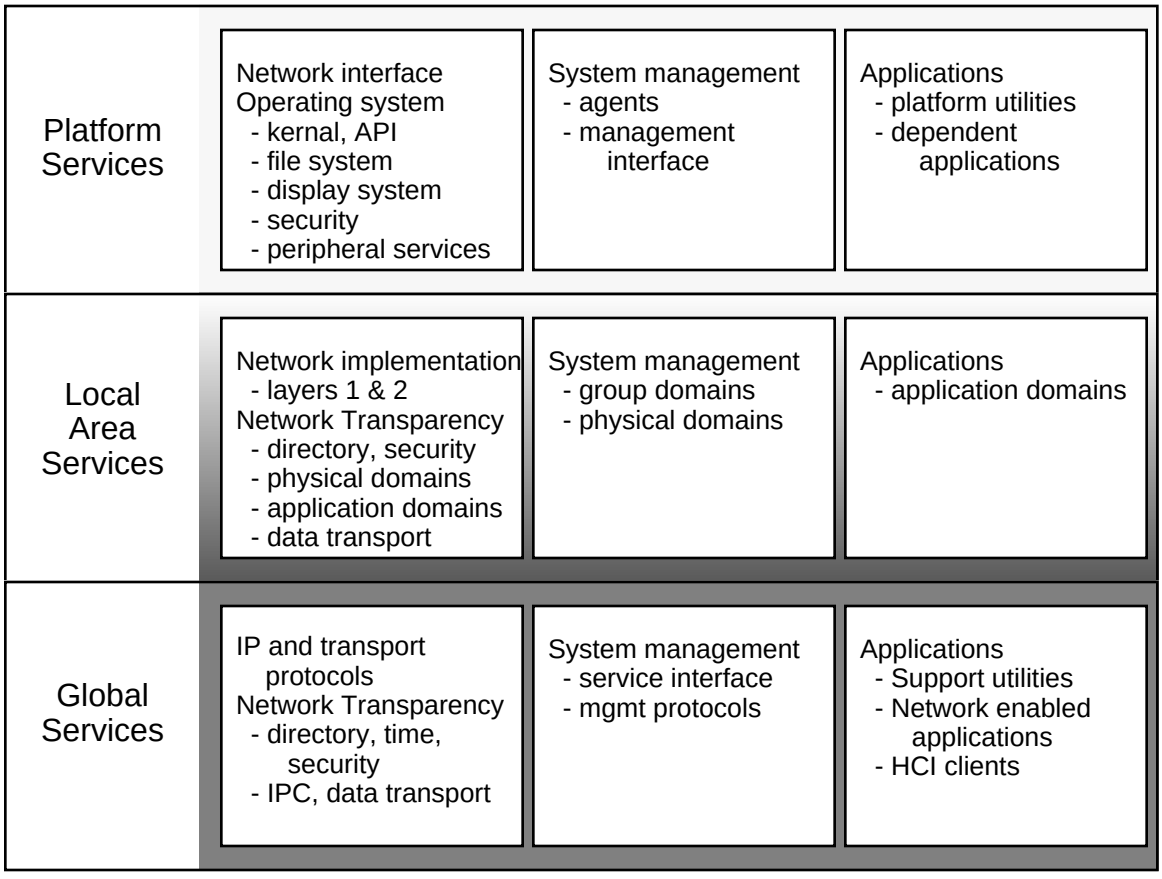

Figure 2-4. Services Domain View

While some types of services are generically and universally constrained to a single domain, more commonly there are options to be selected for a given system instantiation. This ability to choose the domain for a selected service and to have it integrate into the overall system is one of the features providing flexibility in system instantiation. It allows isolation of expensive or complex capabilities to limited domains to minimize cost and risk impacts to the system. Conversely, provision of common global services is a mechanism to support operational changes in the allocation of activities to differing subsystems in the component view, easing adaptation to changing mission needs as well as differences between mission systems. In the following sections the approaches for implementing these services are explained.

2.2.3.1 Network Transparency

Network transparency services are application-to-application services and are an integration of components from all layers of the architecture, as shown in Figure 2-5, where the service is portrayed as the heavy line connecting the two applications. It is important to recognize the integration aspect, as all layers are needed for a successful service.

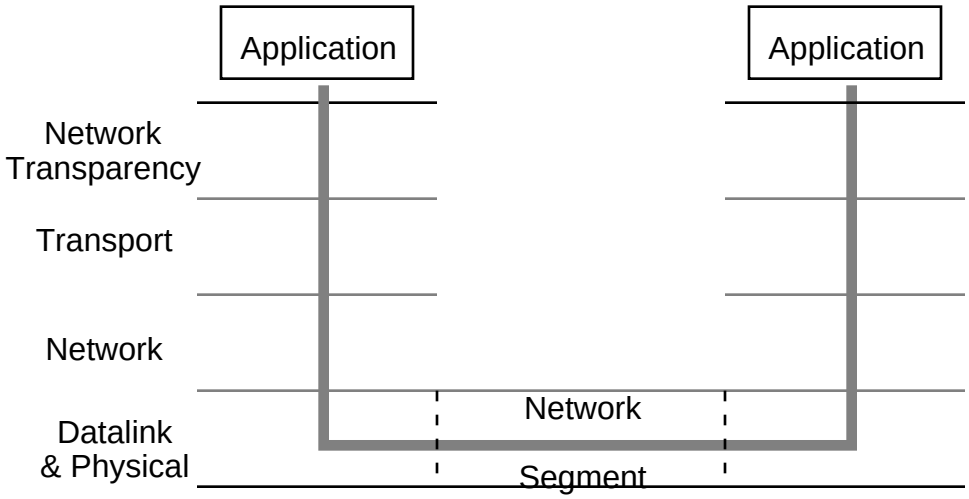

Figure 2-5. Network Transparency Service Architecture

Some services will be provided to all nodes of the network, but others may be restricted to subnetworks because of limits in need and cost of service provision. The relation between the services and the lower layers is particularly important for demanding services. Some services typical of the space mission systems are listed in Table 2-1. For space systems, the data stream services are frequently used for spacecraft management data, and the requirements for reliability, low latency, and bandwidth can be major drivers for the specific characteristics of these lower layers.

The characteristics of the service map directly to the implementations within each layer. For the example data stream service, the combination of reliability and input-output relation require reliable multicast, a linkage of TCP at the transport layer, with a multicast service at the network transparency layer. If the reliability is not absolute, but given as a loss or error rate, then other

approaches may be considered. Linked to this are the error rates inherent in the network segments at the datalink and physical level, and the interfaces between segments. The bandwidth of the low-level network needs to be related to the sum of such services across the network segments. This can lead to complex arrangements for scheduling and costing services or the use of new dynamic allocation approaches, such as RSVP.

Table 2-1. Example Network Transparency Services

Service	Character	Connect Level	Reliability	Input Output Relation
Data Stream	Open ended Low latency	Session	High	1:N
Message - Publish/subscribe	Closed, small unit Low latency	Session	High	1:N
Dataset transport	Closed, large unit	Unit	High	1:1

The architecture resulting for such data stream services generally requires a message server hierarchy, where the messages are passed with very low internal latency in the servers. The hierarchy includes a server set at the global level and another at the local area level. The hierarchy exists primarily because of the need to support heterogeneous needs at the local level, while using the global level for a uniform distribution mechanism. Typically, the global level servers must provide multicasting of data, high reliability, and support for wide area transport. Local servers can vary more in technical performance needs to optimize cost-effectiveness of the solution. Integration of the local and global layers is another part of network transparency to be hidden from the applications.

Similarly, security components will be mapped in a hierarchy. These are a global level ensuring system wide security integration and differing local area implementations mapped to the local area requirements. Typically, local areas with spacecraft control functions and mission data archive elements will be treated with the highest security to prevent loss of critical assets, while areas with public data interchange will be optimized for ease of access, with protection primarily aimed at avoiding system nonavailability.

Typically, network transparency services will be implemented using both distributed software elements to integrate all application usage, and separate support server applications such as a message server and an authentication server. These may operate on separate computer platforms, based on performance, security or configuration management issues.

2.2.3.2 System Management

System management is also perceived as a hierarchy of components with differing domains. At the global level are the services providing an integrated view of the entire mission system for performance and configuration management. At the local area level are the components

corresponding to management by an owner organization. This is essential because there will generally be shared ownership of global network resources. Wide area service will be leased from a commercial service provider. Differing LAN Cells will be owned by NASA, by the customers, and by other organizations. Management will need to be a distributed support application, integrated as a part of the network transparency capability.

Management agents serving to interface the component to the global management API and distributing the data collection and analysis activity will be at the platform and network-attached resource level. Distributed management systems allow for multiple management consoles as well as agent-activated management. These are still somewhat new, although the Tivoli system has been around for several years. Most major vendors are now introducing implementations of at least some distributed management features.

2.2.3.3 Applications

Applications can be configured to any of the three service domains.

1. They can operate as global entities, interfacing to the network transparency API or as platform domain entities interfaced to the platform API.

2. They can also be configured as local area entities, operating as clusters of applications in a domain built around a common interface that serves unique purposes within the application cluster, e.g., peer-to-peer communications. The application domains currently exist, and their use is accepted as providing value without any severe problems because they can be isolated from other parts of the system.

3. Finally, some applications, at a minimum the operating system utilities, are truly tied to platform resources and services. These are uniquely platform domain entities.

2.3 Architecture Changes across System Phases

Figures 2-6 through 2-9 illustrate the evolution of the system configuration in the component interconnection view. This view most clearly shows the changes during the evolution of the mission system during its life cycle.

Figure 2-6 shows a configuration for board-level testing. The configuration consists of a single platform connected to a test apparatus that provides the test environment for the board being tested. The connection between the platform and the test apparatus is through Data Interface Cards. These may be a network connection, but the apparatus may also be attached as a peripheral to the platform. This structure provides a very simple environment for testing at the lowest level.

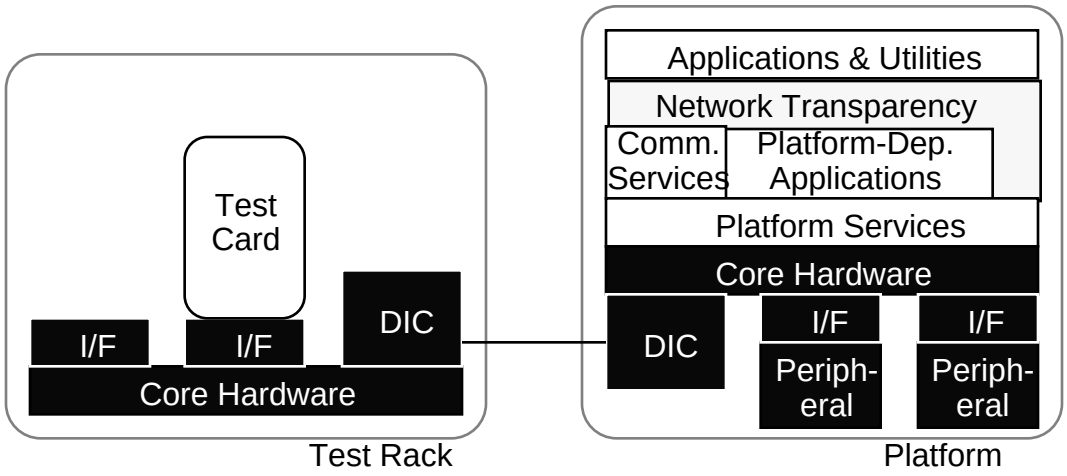

Figure 2-6. Board Test Configuration

Figure 2-7 expands the test environment to address test of subsystems of the operational systems, e.g., spacecraft subsystems. Generally, there are multiple elements being tested and testing may usefully involve multiple persons as the test team. In this case, the test apparatus is definitely extended to provide a connection to a network. The platforms in use connect through the network to the test equipment. Note that the platform internal configuration has no significant changes, but the test rack has added components to support a more complex test environment, including a network interface.

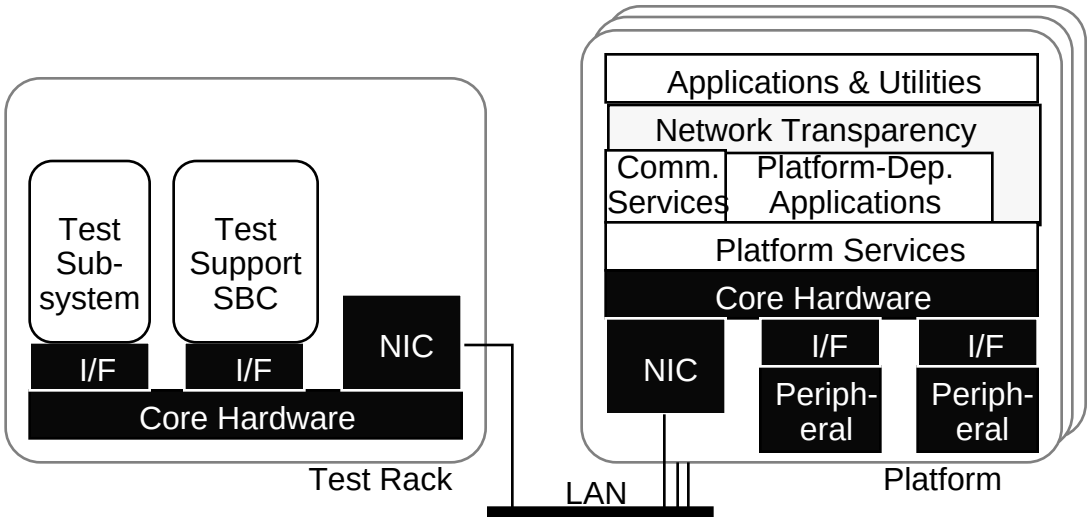

Figure 2-7. Subsystem Test Configuration

In Figure 2-8, the extension to provide an extended environment for integration testing is illustrated. In this diagram, the local networks have been abstracted to subsystems, and many of the expected integration test components are shown. In particular, components for

communication network test support, operations tests, instrument test and unique spacecraft test support are all shown as separate subsystems. In specific cases these may be combined, but those shown are common in current practice.

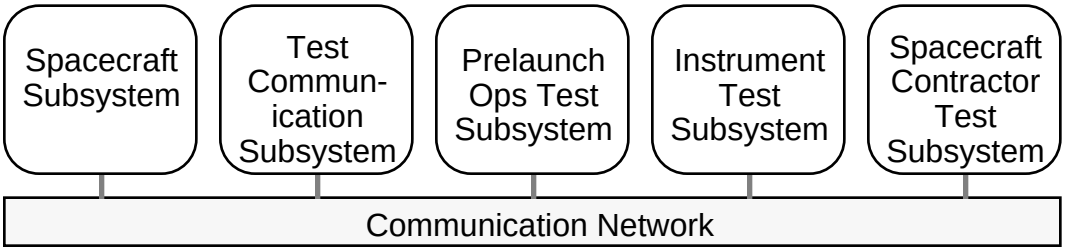

Figure 2-8. Integration Test Configuration

Each subsystem is integrated into the overall test system by the underlying communication network. In practice, many of the test system elements are preliminary versions of the operational components. The spacecraft contractor test subsystem is intended as the analysis subsystem for the contractor to test the spacecraft subsystems for proper operation.

Figure 2-9 illustrates the final operational configuration. In this figure, specific subsystems are labeled for illustration but do not imply that these are the only such options. The support subsystem is intended to show an organization that provides expert contingency support and is generally remotely located from the operational sites. This may be an approach by which MO&DSD support is used by a mission. Any or all of the middle three operational components can be owned and operated by the mission customer or may be shared resources. In particular elements of the communication subsystem and the communication network may be shared or owned either by NASA or commercial service providers.

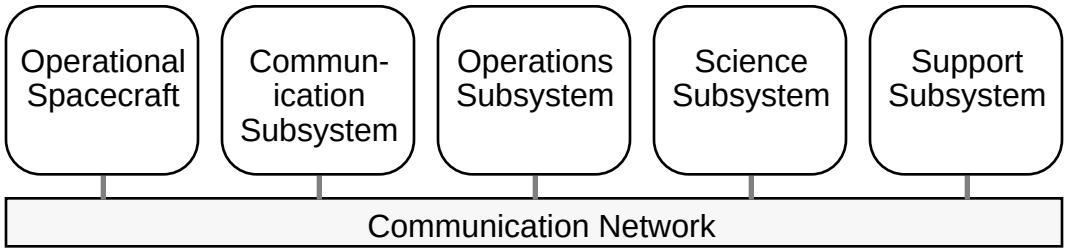

Figure 2-9. Operational Configuration

The primary arena of change over the mission life cycle is in the applications. In the test environment, the operation is primarily real-time, with on-line test analysis and control. People are usually monitoring the test during its performance. In later testing stages, and particularly in operations, the move is towards more automation of the communications and detailed operations, with off-line analysis and planning. As a result, additional analysis and support tools are added, and some of the set environment becomes unnecessary. Additional changes may be made to achieve greater cost-effectiveness of the operational configuration. This can include changes in

the network transparency implementation to optimize for the change from small, local configurations to extended system configurations and to reduce operations costs for routine communications and operations activities.

2.4 External Interfaces and Standards

An external is defined as anything that is not provided as part of the mission system. The essential aspect is that the element is not constructed to match the mission system design and implementation. Typical external facilities are shown in the context diagram of Figure 2-1. External interfaces are typically provided to enable access to shared and contingency resources. Interface types may include data interfaces, schedule interfaces, operational messages, and management interfaces. An interface could range from a dynamic real-time message interface to an off-line exchange of files.

Examples of data exchange through external interfaces include:

- Data product publication to outside customers
- Command and telemetry flow through the Deep Space Network (DSN), the Space Network (SN) or other NASA space-ground communication network.
- Resource management information exchange with NASA space-ground networks
- Use of public WAN services for all external data transport
- Support of I&T with the spacecraft at a contractor facility
- Exchange of diagnostic data with Centers of Excellence, such as flight dynamics
- Instrumentation command and control using shared I&T resources, e.g., thermal-vacuum chambers
- Gathering standard data from outside sources (time signals, star catalogs, solar system ephemerides, geological references)

Within the architecture, external interfaces can be defined within several levels. The intent is that gateway processes exist to map external access to standard architecture transport services, i.e., the server APIs. In the component interconnection view, such gateways most commonly occur as a physical part of the network, e.g., router-gateways to external segments and security gateways but can occur at the platform level, i.e., a connection as a peripheral to a single platform. In the services domain view, an interface service can be at any level. In the layered architecture view, the gateways generally will occur within the network transparency layer and/or the network layer. Given this, the only true interface standards are those associated with the IP suite. As a practical matter, other choices should be constrained by the cost of creating a new interface. Some de facto interface standards are identified in Table 2-2.

Table 2-2. External Interface Standards

Interface Data Type	Interface Standard
Real Time Return Telemetry	CCSDS
Real Time Forward Telemetry	CCSDS COP-1
Bulk Transfer Return Telemetry	FTP
System Management	SNMP
Product Data	CCSDS SFDU
Data Set Transfer	FTP
Document	HTML
Image	JPEG
Video	MPEG

2.5 Mission Configuration

The architecture offers missions multiple modes of adaptation to meet the specific mission requirements over the mission life cycle. Each cell may be scaled by selection of hardware and applications. A specific mission system may be implemented with one or more cells. The number of cells and the hardware and applications may vary over the life cycle. Differing components may be chosen to support multiple spacecraft, or a single spacecraft mission.

The layered domain approach supports changes to the configuration over the life cycle with minimal cost impact. A mission need not implement the WAN layer or all of the LAN layer during the integration and test phase when direct control is possible. The same set of command and telemetry processing applications may then be used over the WAN during the operations phase.

Systems will have standard interfaces to shared resources, such as NASA communications networks. Applications can interface through the platform to special-purpose hardware as required, e.g., during spacecraft I&T. Interfaces to special ground station equipment are another example.

The architecture allows the use of subdomains within the cell that have unique interfaces, e.g., applications that use a unique server or unique peer-to-peer application interfaces. Subdomains may be used to take advantage of existing off-the-shelf components.

For example, a mission using shared resources or serving distributed customers might implement separate cells for mission operations, payload analysis, and so forth all interconnected by the WAN. The primary WAN may be implemented using both private and commercial carrier services. The ground station may be a cell within this architecture, with applications to control the ground station operations and equipment remotely or may be an external system. An illustration of a typical mission integrated into the NASA environment is shown in Figure 2-10.

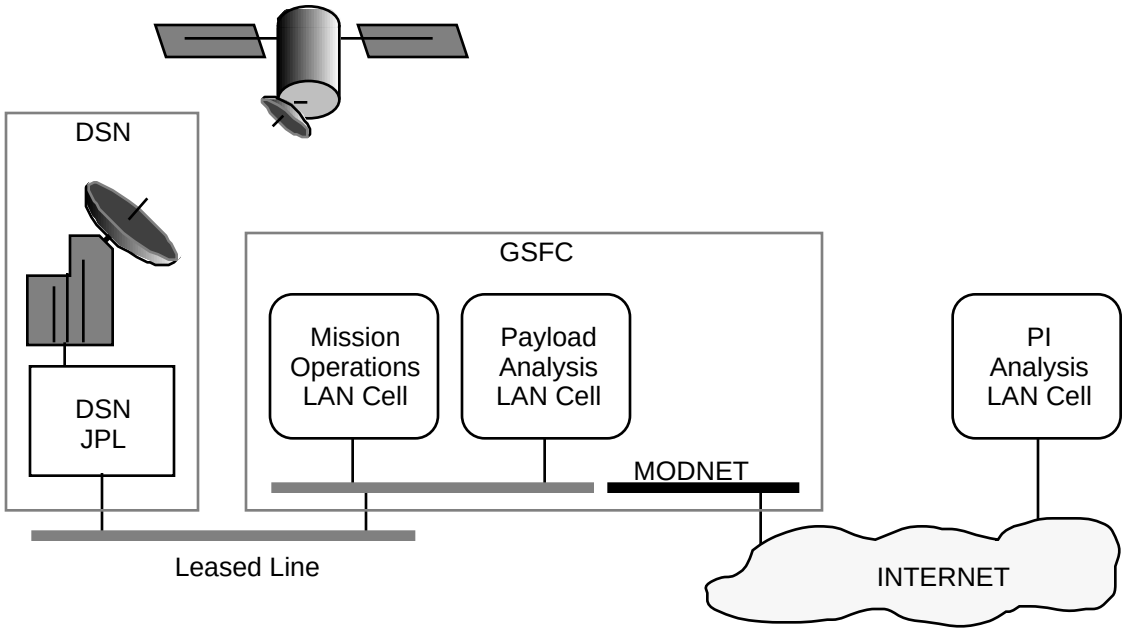

Figure 2-10. NASA-Integrated Mission Configuration

In this approach, the architecture supports evolution over the mission life cycle through the addition, deletion, and expansion, contraction and/or reconfiguration of cell functionality. For example, as the spacecraft and instruments are developed, one integration and test cell will exist. Additional cells to support the functions of mission operations, payload analysis, and Primary Investigator activities will be added before launch, as well as increased I&T capability to support overall system integration. The system will have maximum capability during launch and early operations, and this will decrease as the system becomes stable and as operations become routine.

By way of contrast, a mission with a fully dedicated system supporting all the required functionality over the entire life cycle can be implemented as a single cell, without any WAN services required. Figure 2-11 illustrates such an approach for a mission in which the entire system is self-contained, and the mission is autonomous with respect to NASA institutions. Generally, this would be most useful when the ground system components are co-located. Even the spacecraft could be included in this, if the communication services meet the general needs.

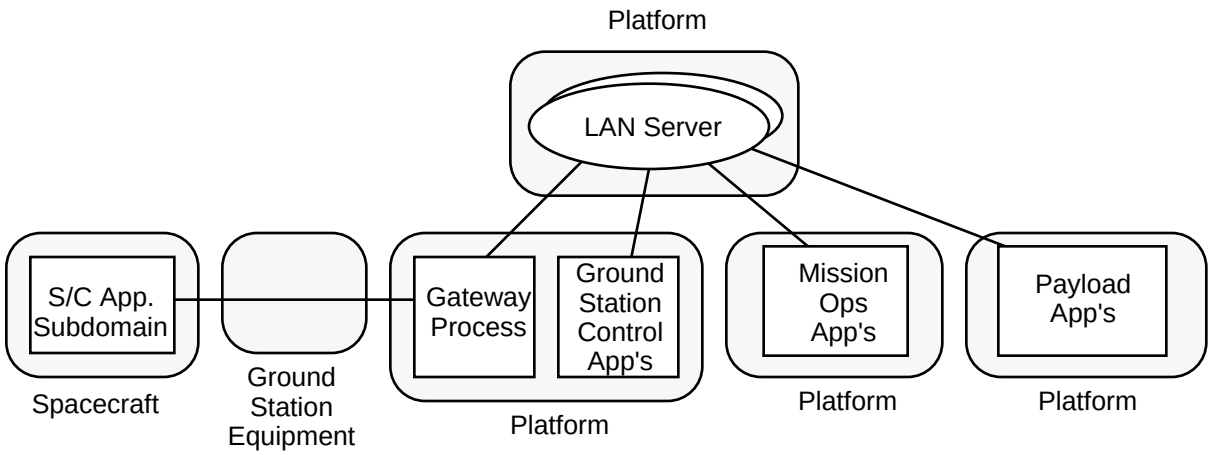

Figure 2-11. Autonomous Mission Cell

In this example, the mission size would vary over the mission phases by changing the number of platforms, LAN server processes and applications over the mission life. For example, the size would be small during spacecraft I&T, peak during launch and early orbit checkout, and shrink again for normal operations as contingencies decrease and automation is increased.

A specific mission implementation may support a single spacecraft or multiple spacecraft. A single spacecraft implementation may use any of the identified off-the-shelf components. A multiple-spacecraft implementation may consist either of connected strings, one for each mission, or it may be built using software with inherent multispacecraft capabilities. For example, some off-the-shelf command and control applications inherently support context switching based on spacecraft ID. This design choice will lead to different choices of off-the-shelf components in the LAN Cell primarily at the application level, and changing course after implementation will be costly. A multispacecraft design will be less expensive if the spacecraft are sufficiently similar in implementation and operations. Otherwise, separate cells directly supporting each variant are likely to be the better approach.

A final variant should be noted. A three-tier approach, with the human operator interface applications separated from the other functional applications, would allow provision of a single cell with all human operations in common, while supporting spacecraft specific cells at any location. This architecture is not yet commonly available off-the-shelf but is increasing in use.

The architecture is based on avoidance of true multimission implementations, i.e., a system that operates widely varying missions. This is because providing such a common system will be either very expensive because of the broad range of required components and operational differences or will impose significant constraints on the missions. If the missions are significantly different, e.g., different spacecraft and/or mission objectives, a multimission approach will also suffer in operations cost since separate operations teams will still be needed.

System management is always integrated at the mission level. System Management has a full mission view of all domains through the standard Management Information Bases and maintains control over the resources that it owns.

Section 3. Software Architecture

This section defines the generic architecture for the software elements. The software architecture mirrors many aspects of the system architecture. It is based on the concept of distributed software components acting collaboratively to provide the needed system capabilities, with the component relationships established dynamically. The software architecture parallels the system architecture. The layered architecture model is used as a framework in which more detailed component interconnect representations are provided. The layered software architecture is shown as a part of the layered system architecture in Figure 2-3.

The layered software architecture view is simplified into three layers: the services layer, composed of the communications stack (OSI reference levels 2-4) and the platform stack; the network transparency layer (OSI reference levels 5 and 6); and the applications layer. The service domain and component interconnection architectures are defined for each layer separately, with interlayer interfaces described in the lower of the connecting layers. Section 3.1 describes the architecture for the services layer, Section 3.2 the network transparency layer, and Section 3.3 the applications layer.

3.1 Services Layer

As noted, the services layer is composed of two stacks. In the NIST reference model, communications services are represented as a platform service. Because of the importance in this architecture of the distributed services, the communications and the other platform services are separated and point to services and resources not local. Otherwise, this representation is the same as the NIST model.

3.1.1 Communications Stack

The baseline architecture for the communications stack is a straight IP architecture. This is a layered communications architecture that generally maps well into the OSI reference model. Figure 3-1 illustrates the primary software interconnections for this stack. Essentially, the software can be divided into the three OSI layers to which software is applied: the transport, network and data link layers.

At the top, the communications stack interfaces to the Network Transparency layer and at the bottom to the network hardware layer. The interface to the Network Transparency layer is used for connection management and exchange of messages. Choice of the transport mechanism is the most prominent part of connection management. The messages are the combined content to and from the Network Transparency and applications levels, as logical messages.

The data transport level performs the conversion between message and transport data packets. Included with this may be the addition of packets to manage message transfer. For TCP, additional functionality and packet exchange are included to perform reliable message exchange. TCP is essentially a synchronous process, accomplished by a series of exchanges between

transport components on either end to verify or disclaim successful transfer of frames, and to retransmit frames when needed. For UDP, message datagrams are created as independent entities and sent asynchronously with no other verification process. Likewise, UDP receives datagrams with no warning or control. There is no added process of constructing or deconstructing messages into datagrams. Both TCP and UDP use a cyclical rendundancy check (CRC) to identify bad packets, and further processing of bad packets is rejected. Note that bad packets are usually missing chunks because of the behavior of lower level elements.

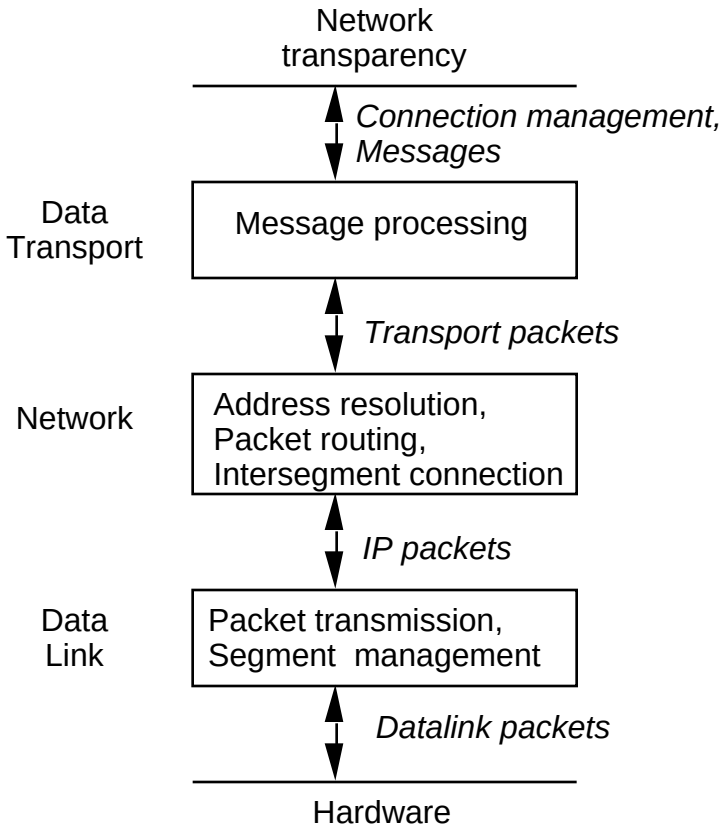

Figure 3-1. Communications Stack Interconnectivity

The network layer performs packet address resolution and routing to the specified address and intersegment connection, providing a common medium between dissimilar datalink network segments. Network address resolution usually uses the support services of the Domain Name System, a standard federated server approach to global network name-address mapping. The network routing function can be used to prioritize transport of packets and to replicate an incoming packet and forward it to multiple addresses. This latter is called IP multicast.

The datalink layer performs packaging of the IP packets into medium-specific packets and managing the transport of these across the network segment. The implementation is usually closely coupled to the hardware implementation of the segment. Commercial examples include Ethernet and FDDI.

For this architecture, in addition to the standard commercial datalink levels, there is the need to address the space communications segments. Currently, NASA uses CCSDS protocols, which do not map into the OSI model at all. It has been suggested that this may be adapted so that the CCSDS transport format, using VCDUs and Reed-Solomon encoding, can be modified to be used as a datalink protocol, with standard IP layered over it. Other options have been suggested. This is not yet resolved and is closely related to evolution in spacecraft implementation.

It is the recommendation of this architecture that a model using a space-unique datalink protocol be evolved and that IP be used over it. Otherwise, some other level must be found at which information exchange can occur, and use of commercial infrastructure becomes problematic.

3.1.2 Platform Stack

The platform stack provides access to local platform resources: the CPU, memory and platform peripherals. The software architecture for this generally follows the NIST model of an operating system core at the bottom, with platform-specific services on top. Software components accessing the stack can interface with either the services or with the operating system core. The services use the operating system core to implement their services. The mapping of capabilities to the operating system and the platform services is dependent on the platform. The architecture includes the use of heterogeneous platforms, so operating system API and service mappings are not a constant. This is one of the reasons for the separation of the network transparency layer.

Operating system capabilities are expected to include multitasking and multithreading, as well as general and extensible access to peripherals. Additional capabilities are expected to be included in at least some system platforms to support real-time processing, multi-user operations, and security. Also implicit support for the variety of available platform hardware resources, e.g., multiple CPUs and redundant arrays of inexpensive disk (RAID), etc., is required. Limitation of used operating systems to a small number of widely supported products is expected to constrain the general complexity of heterogeneous systems.

Platform services are extensive and need to provide support for the full range of development and operations activities. These services will include:

- Data management
- Data interchange
- Human-Computer Interface
- Graphics
- Multimedia
- Internationalization
- Software and system development
- Security
- System management

3.2 Network Transparency Layer

The network transparency layer provides the essential services for interconnecting two software components across the network. The driver for this is to simplify the interfaces and increase the flexibility and capability of possible interconnections. The intent is to provide this interconnection service with COTS products using standard APIs so that COTS applications and infrastructure are more widely applicable. The objectives can be summarized as:

- Providing system-wide integration: system-wide scope and flexible interconnection to meet system needs
- Providing simplicity in interconnection: connection abstraction to simplify interfaces and a simplified implementation of the service infrastructure
- Providing standardization: multivendor standard application interfaces and system infrastructure

3.2.1 Transparency Model

Network transparency architecture follows a general model, which establishes the core capabilities and the interconnection structure of the components. Both global and local area domain capabilities are included components of the model.

The capabilities provided by the network transparency layer include the following:

- Management of interprocess communication, including connection opening, closing and error handling; and transaction management
- Management of process availability, including configuration management, and both dynamic and persistent knowledge of process status
- Support for multiple data transfer types, including datagrams, data sets and data streams
- Support for technical performance, including reliability, latency and throughput
- Provide security for access and resource interaction
- Heterogeneous platform support
- Process usage management, including license authorization and usage monitoring

Since the implemented systems will vary in needed capabilities, size and complexity, not all capabilities for network transparency will be needed for all systems. Additionally, the possible COTS implementations vary widely. The result is that implementation of the layer does not always follow exactly the same architecture.

The layer does usually follow a standard general structure, however, which is illustrated in Figure 3-2. This architecture consists of a software component that provides the basic API services to the applications, and provides the connections to the communications layer and to support services that enable the effectiveness of the connection service.

The connection management element usually connects through the communications layer to another instantiation of the network transparency but may perform such interconnection locally if both applications are on the same platform. The support service components connect to either the local platform services components or to other components through the communications layer.

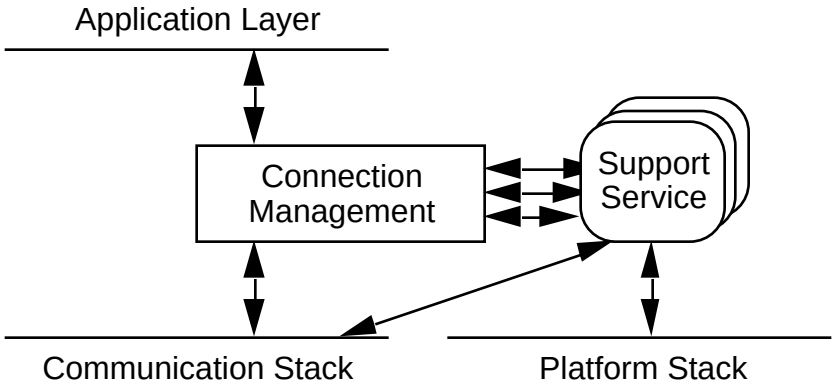

Figure 3-2. General Architecture for Network Transparency

There is a somewhat fine distinction between the support service components in this layer, the components in the platform stack below, and the applications layer above. The difference is that these components provide support capabilities only in service of network transparency and are not services the would be accessed by applications through platform-specific interfaces.

3.2.2 Network Transparency Implementations

This section illustrates some of the most widely used implementations of network transparency capabilities. These are used to show the variety in design for the implementations.

3.2.2.1 World Wide Web

The World Wide Web is an Internet application, rapidly becoming extremely popular because its client browsers provide very user-friendly access to the Internet. It shows both the similarities and differences from the general model. As shown in Figure 3-3, the primary Web components are the HTTP/HTML interface, the Web Server, and the Common Gateway Interface.

The HTTP/HTML interface matches the general model, providing a network transparency API for client applications (browsers in Web terminology). The interface connects an application through the communications layer to the Web Server that performs information management services for the client.

The Web Server can access platform stack services to load Web "pages," i.e., HTML-based documents. It can also access other local applications or platform services through the Common Gateway Interface. For such interfaces, the Web Server provides translation and data transfer management for access between the client and the local application. Current examples include

relational database management systems and specialized search engines, which provide selection of Web information sources through queries.

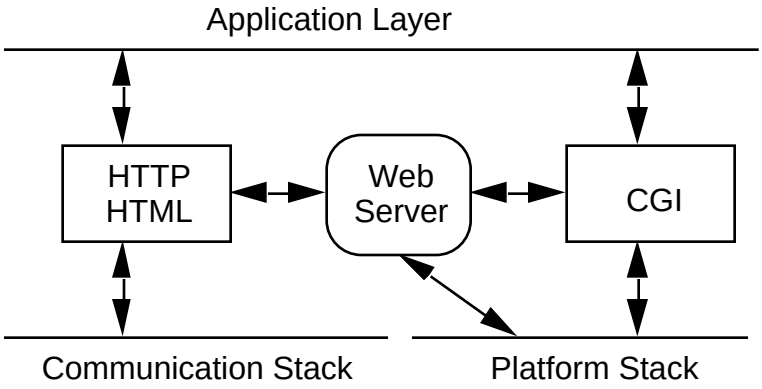

Figure 3-3. Web Implementation

The Common Gateway Interface does not entirely match the general model, as it is intended to be a local interface only and not through the communications services. All network transparency is through the HTTP/HTML interface.

3.2.2.2 Distributed Computing Environment (DCE)

DCE was developed by the Open Software Foundation (OSF), and subsequently incorporated into X/Open. It was designed as a part of a general network transparency environment. The other part of the general environment, the Distributed Management Environment, has seemingly been stillborn. DCE is a commonly available environment, provided by most UNIX vendors and also is available in a Windows environment.

The implementation of DCE is shown in Figure 3-4. Again there are similarities and differences from the simple, general model. The Interface Definition Language (IDL) / Remote Procedure Call (RPC) interface provides general network-wide interface corresponding to the general model. The RPC mechanism provides a synchronous connection between two application components across the network. In doing this it uses two support services, one for Name services and one for security services. The security server also has a separate interface, the Login interface used by principals to authenticate their identification and establish keys for access. These keys are then used with the RPC interface to ensure authenticated access to network resources.

Additionally, DCE provides two services that interconnect the platform stacks across the network. The Distributed File System is a client-server service providing a unified, multiplatform file service. Similarly, the Distributed Time System is a client-server component that provides a uniform system-clock time across the entire network. An absolute time source is used for time baseline, and this is then distributed across the network to clients that update the local platform system clock. Very accurate time distribution is possible with this system.

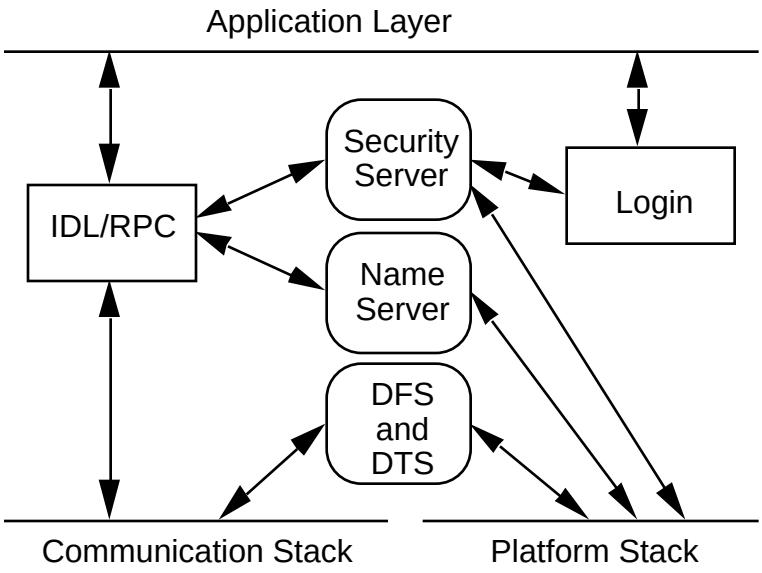

Figure 3-4. DCE Implementation

3.2.2.3 Publish-Subscribe Message Systems

A variety of publish-subscribe message distribution systems have been implemented. They all use the same basic design, as shown in Figure 3-5. This shows the basic API component that connects to one or more message servers for supported message distribution. The basic model is one in which there is a data group established that consists of a message class, a data publisher for that message class, a set of servers and a set of subscriber clients for the class. Whenever the publisher provides a new message, the servers distribute it to the established clients. Transmission throughput, latency and reliability vary widely in the current implementations.

The normal service domain of the message servers is global. However, the implementation can be widely varying, depending on the needed reliability, throughput, etc. Further, for a system with heterogeneous LAN Cells, a homogeneous message server design may not be feasible. In this case, the result is a hierarchical model, with global servers, and, for at least some Cells, a local area domain message server set. Additionally, with current implementations, some applications provide their own domain environment, including a server approach, as shown in Figure 3-6. These are appropriate when the application set is highly valuable and the entire environment is COTS. This approach is some times needed for security reasons as well. It should be avoided unless the value is high because of its greater complexity and probable greater cost.

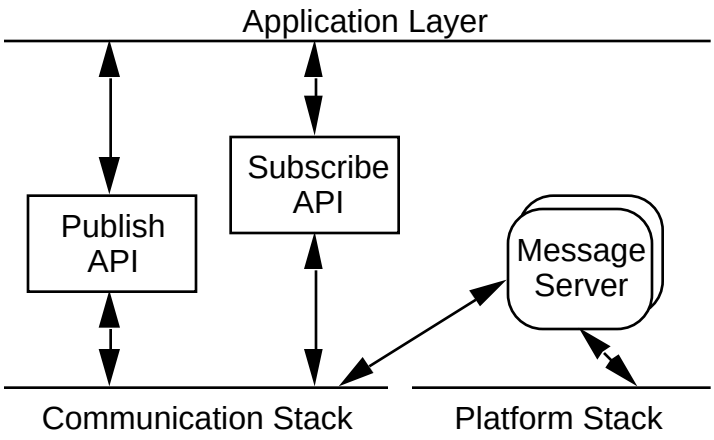

Figure 3-5. Publish-Subscribe Implementation

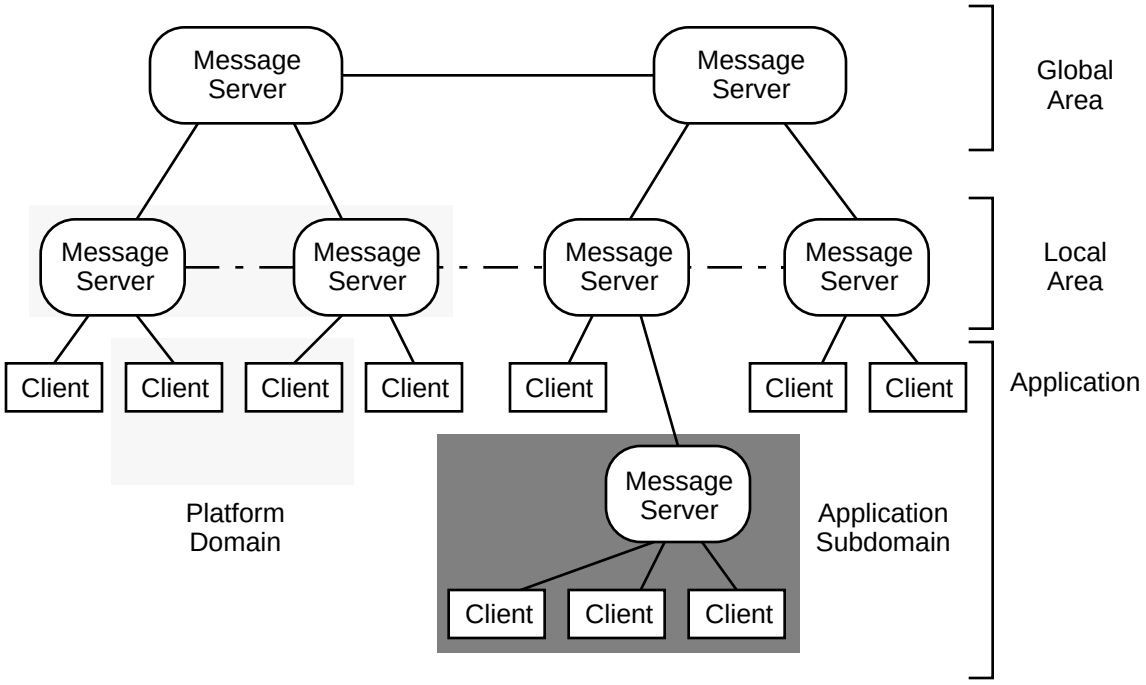

Figure 3-6. Message Server Hierarchical Implementation

3.2.2.4 Common Object Request Broker Architecture (CORBA)

The CORBA approach closely matches our general model, as shown in Figure 3-7. The client and server APIs are relationship oriented, and both can be invoked by a single application component. The client interface is used by an initiating component requesting a service. The server API is used when the component is constructed to respond to such requests for service. All such requests are mediated by the Object Request Broker. Through the Object Request

Broker, Object Services can be invoked for support of either the broker or an application. Object Services include, by design, all the capabilities identified with network transparency. In implementation, however, not all are available. Current implementations are immature and do not include important security, reliability and performance characteristics. Some of the service specifications are very new, and implementations will not exist until later this year.

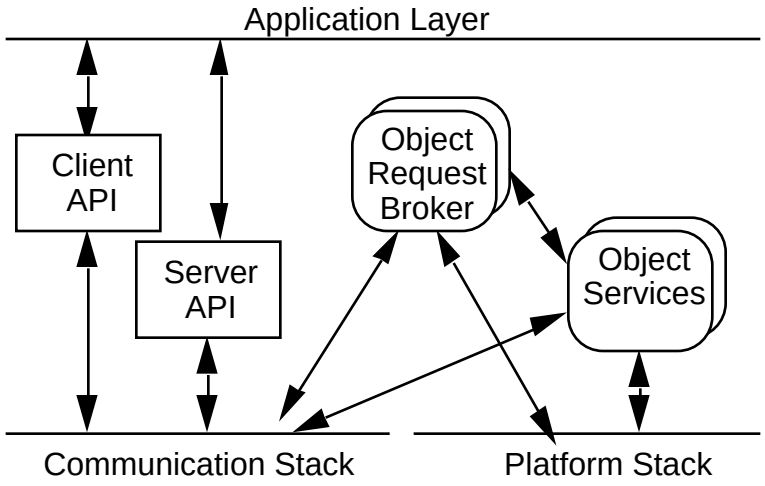

Figure 3-7. CORBA Implementation

The domain of an Object Request Broker is global in concept. However, heterogeneity is a reality, and a broker-to-broker interface API has been defined for integration in such a heterogeneous environment. In such a case, the broker domain is logically local area, and this may also be true in the network sense. This case is similar to the message server case above.

3.2.2.5 Java Implementation

The Java language itself does not provide much in the way of network transparency. However, through the addition of the run-time environment and connections to CORBA, some important features are added. The primary addition is the run-time environment found in Sun's HotJava or other Web browsers, which provides a common run-time environment for compiled Java applications. This creates the ability of compiled applications to run on any platform with the run-time environment, regardless of operating system. When added to a distribution process and integrated with CORBA, there is the potential for truly dynamic distribution and integration of applications. This area is one that will require close monitoring to take advantage of features as they become widely supported.

3.3 Application Layer

The architecture of the applications responds to the range of requirements and uses the capabilities of the infrastructure to provide effective capability. Flexibility of implementation with low cost is a primary focus. In the following sections are presented a summary classification of application requirements and the application architecture that results.

3.3.1 Applications Requirements

Applications for the system are defined to address the three primary elements of system activity: provision of data products to the data customer, system resource management, and mission system development and evolution. The applications capabilities and relationships are discussed further in Section 4.

The primary purpose of the mission system is to acquire and deliver to data customers those data products that satisfy their objectives. Applications provided assure prompt and accurate capture of objectives through the planning process and capture of desired product specifications through the evolution support tools. Product generation and quality analysis tools support preparation of the appropriate products and assessment of any changes needed to meet the customer needs. Data transfer and access servers provide for timely delivery of the data to the customer.

System resource management optimizes use of resources to meet customer goals and objectives. These applications provide four general capabilities:

1. Planning activities to sustain system operations and conserve resources
2. Monitoring of activity to assess current performance and predict future behavior
3. Selecting actions to optimize performance of plans, and respond effectively to unexpected events
4. Effective execution of actions, including sequencing and translation to appropriate component directives

The current architecture emphasizes the commonality of system management for all components in the system, while recognizing the variations. Existing COTS approaches to resource management are becoming more capable and are being extended to management of resources that traditionally have been performed only with custom applications. A common, system-wide view is expected for monitoring, though some shared resources may not be under direct system control. More of this management capability may reside in the network transparency layer.

Creation and evolution of the mission software system throughout the life cycle require application of tools that support change in behavior and automation of activity for cost-effective operations. This capability includes effective maintenance of existing capabilities and evolution of the capabilities themselves. The tools provide four general capabilities:

1. Development of baseline system software objects
2. Capture of baseline system objects, including data structures, component characteristics, and system behavior
3. Evolution of these system objects either in response to changing performance or system reconfiguration throughout the mission life cycle
4. Support for both immediate usage of definitions for flexibility and for performance-optimized compilations of the changes

All three capabilities may require real-time performance for some parts. Typically, such performance needs are isolated to smaller parts of the system to minimize cost impacts. Data product acquisition and delivery may involve very large dataset operations, up to terabytes in some cases. System development and evolution support uniquely emphasizes flexibility in configuration. The operational environment emphasizes reliability and automation to minimize cost. All extend the need for flexibility and ease in configuring system elements for providing capability rapidly and at low cost.

3.3.2 Application Architecture

A multitier client-server model is used to provide the needed flexibility with allowance for by passing intermediate tiers for performance. The concept, usually called "three tier," provides for separation of an information server from the human-computer interface client by a process layer as shown in Figure 3-8. The applications connections are implemented through the network transparency layer underlying the applications. For the space mission environment, the complexity forces us into this model with the recognition that there is really not a single, simple tier in between.

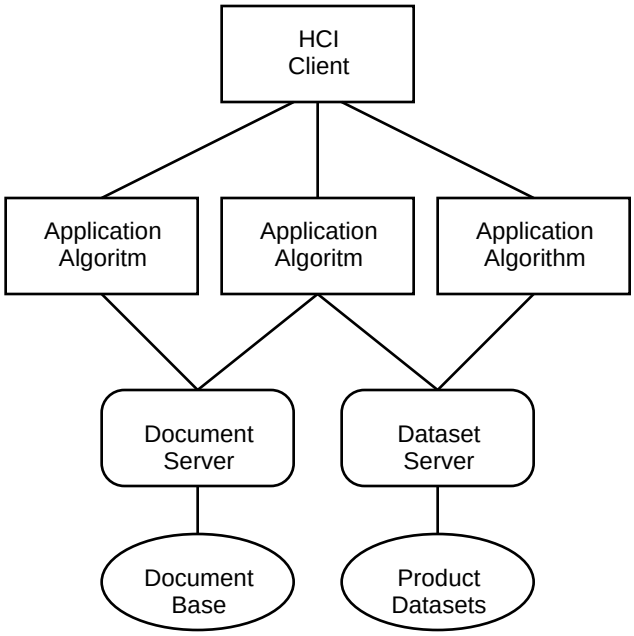

Figure 3-8. Three Tier Client-Server Model

Complications for this simple model arise in several ways. The term information server is used because of the reality that other than standard datasets need to be included, e.g., real-time data streams. The true mission applications need to be formed from a federation of application components that directly interconnect. The need for automation means that any HCI client may be many times removed from contact with a particular application client. Using existing COTS products makes enforcing true client-server potentially very difficult. Finally, the client-server

relationship may not be a permanent aspect of a specific component but only a transitory relationship, supporting more of a peer-to-peer component status.

The information server class of application is expanding quite rapidly. For the case of real-time data collected from an apparatus, the server is the application that collects the data from the apparatus and makes it available for other application elements. In the publish-subscribe model, the server is the publisher of data. The distribution of the data to other elements is part of network transparency and not considered here as an application element. Other forms of information bases include all the forms of databases and file systems and now also include object repositories for persistent object storage.

The mission applications are frequently so complex that they are and will be constructed from multiple components, which need to be allowed distribution for technical performance and cost factors. Given this, there needs to be an application construct that structures the integration of the set of components. There are two models for this: the transaction model and the workflow model. In the transaction model, there is a specific application containing the integrated definition, which controls the activation of the needed elements through the services of a network transparency layer transaction monitor. The transaction model is particularly useful when there is a high level of repetition in the processing, i.e., many occurrences of the same sequence with dynamic variations in the time of occurrence. Transaction monitors are highly developed to make this an efficient activity. The workflow model is more object oriented, in that the initial definition of the activity includes the initial data and the specification of the processes and sequence of events defining the resulting product. It is particularly suited to circumstances where similar but varying sequences must occur or which have no driving need for computer efficiency and low latency in sequence processing.

In the expected operational environment, the HCI client is not directly attached to most of the active processes. People enter into the activity only for general planning and management and for exceptions processing. For the applications this means that many will have no connection to an HCI. For the HCI client this means that it is attached to the person, not an application, and needs to be very flexible in its ability to enable the human interaction. One of the better current examples of this is the Web browser/client, which supports text, graphics and multimedia data interactions but has no specific application connection.

Existing COTS products do not generally conform to a simple client-server model. Many were designed as two-tier client-server, with the HCI and the algorithm processing in a single component. Others conform to no standard client-server model and may be monolithic, stand alone applications. Where feasible, COTS applications should be integrated in as client-server applications and support a common HCI interface to reduce operations cost. When such an integration is too costly, then exceptions will have to be accepted.

Status as a client or server is not always a permanent state. The intent is to support peer-to-peer relationship by use of client-server contact relationships. Specifically, two components can connect in a client-server contact in which one is the initiator and the other responds. In a different component connection, there may be a different client or server allocation. Thus, a single component can be a client for one pair and a server in another. These relationships may be time varying or the pairwise relations may be permanent. Because peer-to-peer is more

complex to manage, it should be used only when it enables much higher effectiveness. One potential example is a software agent that gathers information for an HCI client. Another is that of an application that collects information from a variety of sources, then initiates a connection to feed processed results to a different process. In general, an application that serves both roles should have two interfaces, one for the client role and one for the server. With some implementations of network transparency, it must have two interfaces.

The result is that, while following the general three-tier client server model, actual instantiations of the architecture will be more complex. An example is shown in Figure 3-9.

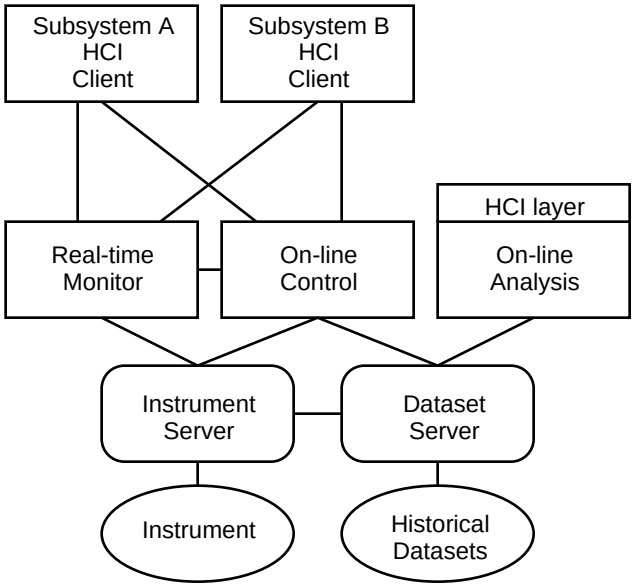

Figure 3-9. Application Architecture Reality

Section 4. Application Design

This section contains the design of the system in terms of the applications expected for this system. The representation is that of the generic applications, their allocation to types of clustered operations centers, and the data interconnection between the applications. The first section describes the general design characteristics desired, the second the overall connectivity, and the third contains the more detailed description of the generic applications and their relation to the other views of the system. This mapping of capabilities to components is not intended to serve as a final answer but to show the capabilities generally needed for completeness in the space mission application area.

4.1 General Design Features

The desired applications design has the following features:

- Common configuration database and system evolution tools
- Automation support and low cost operations
- Low cost implementation
- Low cost operations
- Space-ground interchangeability

Each implementation must trade the desirability of each of these features against each other and against the availability of the feature in off-the-shelf applications.

4.1.1 Common Configuration Database and System Evolution Tools

Two options can be used in providing the configuration database for a mission. Most off-the-shelf components come with a configuration structure that must be initialized and tools for creating it. This structure could be used without modification. However, the structure may not be flexible enough to allow the use of the product for multiple missions or may require duplicating data in several places in the configuration (if different applications need the same data). An alternative is to use a database management system to store the configuration in an essentially vendor-neutral format. Configuration data would be stored in the database using spacecraft identifier as a primary key. As local applications are started, they would be provided with the spacecraft identifier and will access the database to retrieve parameters. For applications that require a flat file interface for setup, a preprocessor will be needed to extract the configuration parameters from the database and builds the flat files automatically.

Using a database for storing configuration information provides a number of advantages:

1. Using a database to store configuration parameters provides the flexibility needed to operate multiple spacecraft from the same operations center. Using the spacecraft

identifier as a key for retrieving data from the database creates limited overhead in implementation. Even in cells that are intended to support a single spacecraft, the implementation overhead does not significantly impact the cost of developing the spacecraft parameters. It does, however, make it easier to reuse a cell for supporting another mission. Since most of the upcoming missions will be part of a series (MIDEX, SMEX, University class), this flexibility will be very important to groups wishing to capture future missions in the series.

2. Using a database also provides a simple interface for modifying configuration data during I&T. Most database management systems provide a simple forms interface that can be used to create data entry and display screens. Using this interface, the spacecraft engineers will be able easily to enter and maintain the database during this phase.

3. Using the database will also ease the transition between I&T and operations. If both phases use the same database structure, the database can be easily migrated into operations. The only necessary changes should be to implement a more stringent configuration control over the database in operations (using security features of the implementing database management system).

4. Using a database provides an independent interface for configuration management so that the system is insulated from changes in any single application, e.g., a data format change.

The disadvantages to using a database are also significant. They are:

1. The development of either "glue" code to allow the extraction of data for flat-file configured applications

2. The development of a database application to create and maintain the database

3. The development of multiple views of the data to satisfy different applications needs for accessing the database

While the creation cost for the database management system approach is significant, many mission systems will find the life cycle cost is lower for this approach. MO&DSD can assist by providing a common mechanism and tools, avoiding full implementation by the mission. Implementation may also be highly supplemented by the DBMS products tools.

4.1.2 Automation Support and Low Cost Operations

A major feature of this design is to allow for cost-effective automation support. The system will be configurable to provide the appropriate level of automation to ensure that the goals of the particular mission phase are met and that the mission is not endangered. Mechanisms for achieving automation are management by exception and the automated ability to notify operations staff when human intervention is required. In management by exception, the operator is presented information about anomalies in the operation of the system. The operator then uses the tools of the system to identify the anomaly and to act on it. With an automated ability to

notify the operations staff, the correct personnel needed to respond to an anomaly could be notified using electronic mail, automated telephone, fax, or beeper.

A major goal of automation is to reduce the operations costs. The design also lowers operations costs by allowing applications to run on any of the platforms in the cell (thus reducing the need to duplicate or triplicate every platform to achieve operational availability).

4.1.3 Low-Cost Implementation

This architecture achieves low-cost implementation through efficient integration of COTS products for most components. By using proven commercial components, the amount of new development will be limited. The use of any particular product must be weighed against the cost of developing "glue" code necessary to interface it with other components. Experience of other parties in creating interfaces when using a common server API has demonstrated efficiency in this application domain.

4.1.4 Space-Ground Interchangeability

Providing the ability to migrate applications developed for the ground system to the spacecraft reduces operations costs by allowing intelligent analysis and control applications to directly affect the spacecraft without the need for human intervention or telemetry downlink/command uplink. This can be accomplished in one of two ways.

The first is to have a high level language that is supported by process servers in both the space and the ground environment. This allows procedures to be developed, tested and used on the ground before being migrated to the spacecraft. This can minimize ground operations as the mission progresses and provide higher capability for sustained autonomous spacecraft operations. The potential drawback to this approach is a decrease in performance caused by the higher level of abstraction.

The second way is to provide an application and infrastructure environment that is common to both the ground and the spacecraft. This requires greater coordination but simplifies reuse of the software in the ground processing system and simulator and still allows applications to migrate from the ground to the spacecraft. Working directly with the flight code maintains higher performance by staying closer to the machine. As space-borne hardware adopts commercial standards, e.g., standard processors, this becomes more feasible.

4.2 Application Interconnection Overview

A summary of the applications is shown in Figure 4-1, the interconnection diagram for the applications. This figure shows the generic applications for the system and the logical connections between applications. Message level interchanges are shown as arrows, while datasets are shown as ovals in the diagram. The collection of applications into the operations center is shown, though these can be separated into more than one center. The other ground system components are indicated by the rounded rectangles and include the development center, the I&T subsystem, the Ground Station element, and the spacecraft. Any of these may have more than one instantiation in an implemented mission system.

The generic applications indicated represent the application building blocks for the second generation architecture. More than one implementation is available for most of these. The boundaries have been selected to correspond to reasonable interfaces and to the existing interfaces in available products. The descriptions of the generic applications and a preliminary mapping of these to known products is presented in the following section.

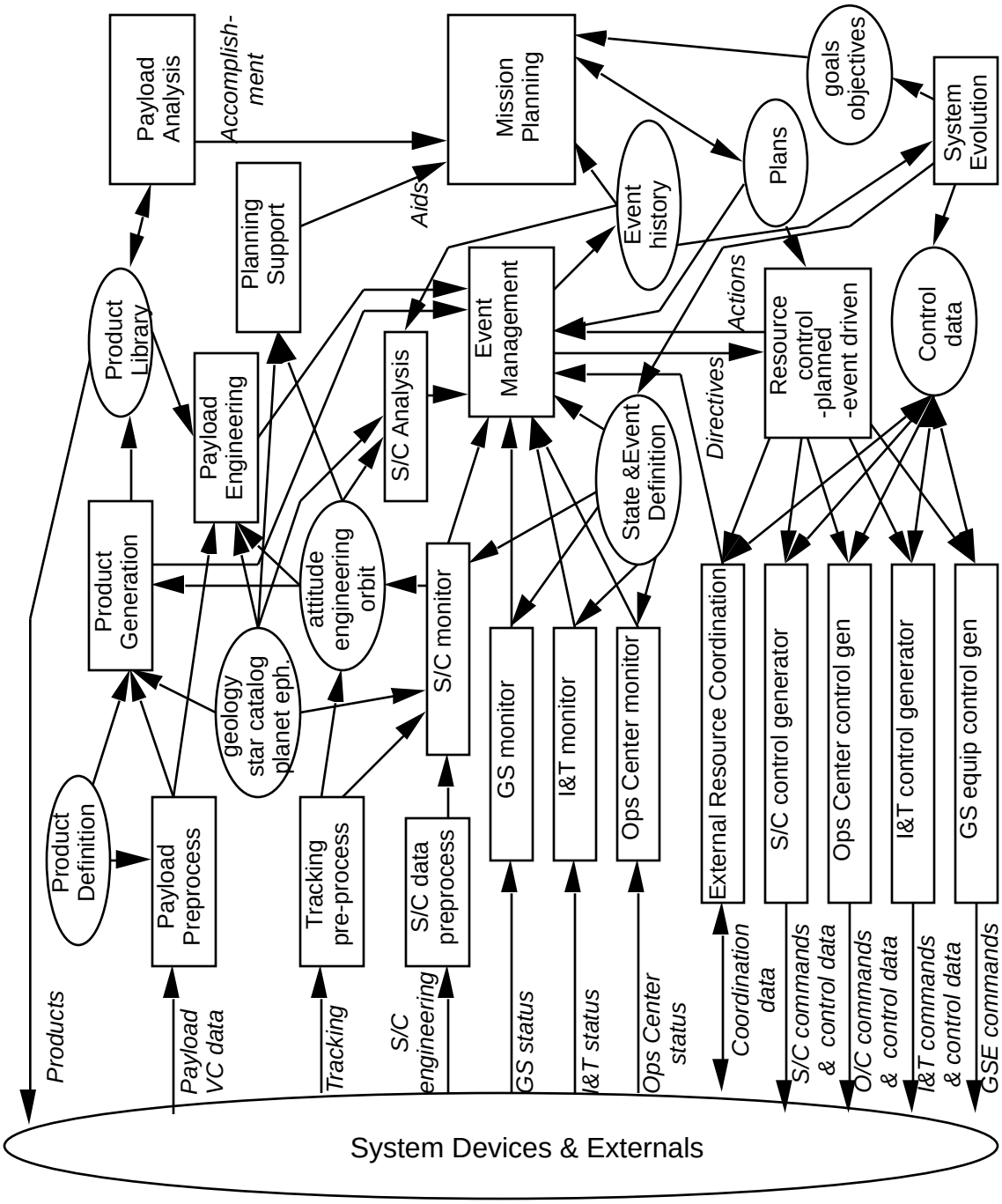

Figure 4-1. Applications Interconnection View

4.3 Application Descriptions

4.3.1 Event Management

The system event management application supports the end-to-end, system-wide management of all the resources required to support the mission, including the ground system equipment, the spacecraft, the I&T system, and the operations center. The application maintains overall knowledge of the state of the system, monitors events from ground system, spacecraft, operations center, and I&T equipment, and issues high-level directives to manage system resources.

Alternatives for implementation include: HP OpenView, Altair Mission Control System (MCS).

4.3.2 Ground System Control Generator

The ground system control application generator generates commands and data to the ground support equipment, including the antenna, RF support equipment, tracking equipment, station data interface, and for remote stations, power, thermal systems. The control generator uses the control repository for access to format information.

It is assumed that Management Information Bases (MIBs) are available or will be developed.

The alternatives include: HP OpenView and SUN Solstice; Manufacturing Message Specification (MMS)-based applications.

4.3.3 Ground System Monitor

The ground system monitor application accepts and displays status data from the ground system equipment, including the antenna, RF support equipment, tracking equipment, station data interface, and for remote stations, the power and thermal status. Status is reported not only at the hardware or platform level but also at the software application level supported by this equipment.

Implementation alternatives will correspond to those selected for the Ground System Control Generator, with common additions such as Altair's MCS.

4.3.4 I&T Control Generator

The I&T control application generates commands for the spacecraft and the I&T support equipment. Spacecraft commands are generated as a result of a user request and built based on information contained in the spacecraft database. Commands are validated and converted into a binary form for transmission to the spacecraft. Commands are output in a bit serial form to the spacecraft baseband input directly or modulated appropriately for input to the antenna, transponder, or subcarrier demodulator depending on what spacecraft ports are available and what is being tested. It may also be required to perform code conversion, level conversion, pseudorandomization, or fill bit insertion on the baseband command stream. Spacecraft commands may be contained in higher level procedures or scripts and may include files or tables to be uploaded to various spacecraft subsystems. This database-driven command generation application is destined to form the basis for the real-time component of the spacecraft control application.

Commands for the I&T support equipment may also be contained in procedures and scripts and will be issued to configure the equipment necessary to communicate with and control the spacecraft and its environment. This may include modulators, upconverters, attenuators, special test harnesses and spacecraft stimulation equipment. The stimulation equipment may be part of a larger spacecraft simulation system. The I&T control application will likely be required to issue commands via a separate serial (RS-232 or RS-422), parallel, or IEEE-488 (GPIB), or IP-based bus. It is also desirable for this application to have the ability to connect directly to isolated components of the spacecraft before installation (or after removal) for test and verification. This may introduce the need for another type of hardware and software interface. For example, it may require a mil standard 1773 (fiber optic version of the 1553) connection that provides command packets to a spacecraft processor.

Appropriate control is provided to restrict access to critical system components or operations, via control of access and control of user privileges.

It is assumed that MIBs are available or will be developed.

The alternatives include: OS/Comet CCL, EPOCH2000.

4.3.5 I&T Monitor

The I&T monitor application accepts telemetry from the spacecraft, receives status information from the I&T support equipment and performs analysis to determine state, generate events, and provide displays. The I&T monitor application typically accepts spacecraft telemetry in a serial bit stream format with a synchronous zero-degree clock. The spacecraft may provide baseband data and/or a modulated RF signal. Accordingly, the I&T support equipment may include hat couplers, attenuators, downconverters, demodulators, bit synchronizers, deinterleavers, convolutional decoders, derandomizers, Reed-Solomon decoders and frame synchronizers. Each of these units may provide status information to the I&T monitor application via a separate serial (RS-232 or RS-422), parallel, or IEEE-488 (GPIB), or IP-based bus interface. The status information is made available to the user via a display and may be checked to ensure the proper configuration is maintained. The spacecraft data are subjected to data logging, telemetry decommutation, engineering unit conversion, limit checking, long-term storage, and distribution. This database and rule-driven telemetry analysis is destined to form the basis for the real-time component of the spacecraft analysis application. It is also desirable for this application to be able to connect directly to isolated components of the spacecraft before installation (or after removal) for test and verification. This may introduce the need for another type of hardware and software interface. For example, it may require a mil standard 1773 (fiber optic version of the 1553) connection that receives source packets from a spacecraft processor.

The alternative integrated applications include: RTWorks, EPOCH2000, and OS/Comet.

4.3.6 Mission Planning

The mission planning application creates an integrated schedule of all required operations center, spacecraft, ground system, and I&T equipment. The analysis tools support the ability to plan future spacecraft events (e.g., maneuvers), payload events, and to identify and schedule the

necessary ground system, spacecraft, operations center, and I&T equipment required to support the spacecraft.

The alternative mission planning applications include: SPIKE, MOPS, and GREAS.

4.3.7 Operations Center Monitor

The operations center monitor accepts and displays status data reported by the operations center equipment and its applications, displays the status and configuration data, monitors performance, and checks for alert conditions such as limit violations. Status is reported not only for the equipment or platform level but also for the applications supported by this equipment.

It is assumed that MIBs are available or will be developed.

The existing alternatives for this type of application include: HP OpenView and SUN Solstice.

4.3.8 Operations Control Generator

The operations control generator application generates commands and data for the operations center hardware. It supports control center reconfigurations. Automated scripts may be triggered by the system resource manager to configure the operations center. The control generator utilizes the control repository for information access.

It is assumed that MIBs are available or will be developed.

The alternatives include: HP OpenView and SUN Solstice.

4.3.9 Payload Analysis

This application performs data analysis on the payload data. It typically resides on an instrument engineer's workstation and may include tools for data visualization and statistical analysis.

Implementation alternatives include: STATLab Pro, PV-Wave.

4.3.10 Payload Engineering

The payload engineering application supports telemetry analysis to maintain the health and safety of the payload (instruments), including diagnosis of anomalies, trending, calibration of instruments, etc. The payload engineering monitor application payload accepts telemetry data after pre-processing and also can obtain payload data products.

Note that the implementation is tailored to meet the needs of the specific mission. One may have a separate payload engineering application, or payload engineering and spacecraft monitoring applications may be combined.

The alternative telemetry applications are the same as those for the spacecraft monitoring application. The alternative data analysis and visualization applications include: PROBE & Patterns, Matlab®, and STATLab Pro.

4.3.11 Payload Pre-process

The payload data pre-process application accepts, accounts for, and sorts frames by VCID. It extracts, accounts for, and routes service data units within the frame (packets, bitstream data units) and routes the data units to the proper application (product generation or payload engineering monitor).

These applications are assumed to be data and table driven via the product definition databases associated with the application.

There are commercially available telemetry processing products that support this function. These include products from Loral Test and Instrumentation, Veda, Decom Systems, Aydin Monitor, Avtec Systems, etc. There are also government off-the-shelf (GOTS) products that support this function including the 521 telemetry processor, the Programmable Telemetry Processor (515/530), Telemetry Processing control Environment (TPCE) and PACOR II.

4.3.12 Planning Support

The Planning Support tool creates planning aids from the primary databases. Typical planning aids include spacecraft state predictions, environmental trend analysis and predicted cost assessment.

Alternative implementations include: Navigator, OASIS, Orion, STATLab Pro.

4.3.13 Product Generation

The product generation application accepts packets, accounts for packets, sorts packets, creates time-ordered files for each spacecraft contact, and distributes the files. The application also creates time-ordered files of packets using customer-defined criteria. The product includes a report of the data quality. Finally, the application accepts data from the attitude and orbit determination systems, packages it, and stores the products. Additional levels of processing, such as engineering unit conversions, instrument calibrations, et al., may also be included for any given mission.

These applications are assumed to be table driven and able to support reconfiguration of tables associated with the application.

The alternate applications are GOTS: TPCE and PACOR-II. The alternative flight dynamics applications include: Satellite Toolkit (STK) and Oasys for orbit determination.

4.3.14 Resource Control

The resource control application accepts the schedules from mission planning and the directives from event management and determines the required directives for the spacecraft, the operations center, the I&T system, and the ground equipment. These include both planned (schedule derived) directives and self-generated event driven directives (e.g., in response to changes in the equipment status). The resource control application reports defined events to event management.

The application alternatives are the Spacecraft Command Language, and Altair MCS.

4.3.15 Spacecraft Data Pre-process

The spacecraft data pre-process application accepts, accounts for, and sorts frames by VCID. It extracts, accounts for, and routes data service units within the frame (packets, bitstream data units) and routes the data units to the proper application (product generation or spacecraft engineering monitor).

These applications are assumed to be table driven via configuration databases associated with the application.

The application alternatives are the similar to those for the payload pre-process application.

4.3.16 Spacecraft Monitor

The spacecraft monitor application generator accepts spacecraft data, after pre-processing, from the spacecraft antenna, RF support, power, thermal, C&DH, attitude control, and orbit control subsystems. It displays the data, checks for limit violations, compares with expected values or states, and provides alerts. Displays include tabular telemetry displays, time plots, gauge displays, and graphic displays of spacecraft subsystems that illustrate spacecraft status.

The integrated applications alternatives include: EPOCH2000, RTWorks, OS/Comet, the STORM IMT/G2 and the Altair MCS, combined with a telemetry processor such as the Loral Test and Integration System LTIS-500, Veda, and others. (The EPOCH2000 contains its own front end application.) Display applications such as DataViews or a Web product may be used to supplement these applications (in some cases it is incorporated within the product).

4.3.17 Spacecraft Analysis

The spacecraft analysis applications support telemetry analysis to maintain the health and safety of the spacecraft and its subsystems, including diagnosis of anomalies, trends, status of expendables, monitoring of maneuvers, etc. Spacecraft analysis updates the System Event Manager with current knowledge of the spacecraft state.

It is expected that several separate applications will be used and integrated via middleware.

The alternative data analysis and visualization are: PROBE & Patterns, Matlab, and STATLab Pro. The alternative flight dynamics applications include STK and Oasys.

4.3.18 Spacecraft Control Generator

The spacecraft control generator application generates commands and loads destined for the spacecraft antenna, spacecraft RF support, C&DH, power, thermal, attitude control, orbit control, and payload subsystems. These functions include user command input, validation, and formatting. The application supports commanding initiated by an operator or an event. Higher-level procedures and scripts may be created, controlled, and executed. These procedures and scripts may be triggered by the system resource manager or the operator. Appropriate control is provided to restrict access to critical system components or operations through control of access and control of user privileges.

The control generator uses the control repository that contains the spacecraft database (command definitions). The uplink of commands over the space-to-ground link (e.g., COP-1) is handled by an application at the ground station.

The alternative applications are the Spacecraft Command Language, Altair MCS, STORM IMT, EPOCH2000, and OS/Comet/CCL.

4.3.19 System Evolution

The System Evolution application provides creation, configuration management and analysis support for defining the system configuration tables. The databases supported through this include Product Definition, State & Event Definition, and Control Data.

Implementation alternatives include DBMS management tools and application-specific tools as noted by the other applications.

4.3.20 Tracking Pre-process

The tracking pre-process application accepts the tracking data, and sorts and formats the data to be compatible with spacecraft analysis and spacecraft monitoring applications.

The RDProc application supports these functions and converts GSFC standard formats into formats compatible with the STK family of products.

4.3.21 System Databases

The following are the primary databases shown in the figure.

4.3.21.1 Attitude, Engineering, Orbit Databases

This database manages the storage and retrieval of the telemetry parameters associated with the attitude, engineering and orbit systems of the spacecraft. It may include the use of database management systems like Oracle or Sybase.

4.3.21.2 Control Database

The control repository contains the data necessary to support control of the spacecraft (command definitions), operations center, I&T equipment, and ground system equipment including conversions to binary and default data values.

Database Management Systems: Oracle, Sybase

4.3.21.3 Event History Database

The event history repository contains an archive of events and spacecraft telemetry. Events include not only spacecraft telemetry and commanding events but also definable events for the integration and test, operations center, and ground equipment. Events are definable and may

include such parameters as name, type, criticality and time of occurrence. This application supports retrieval, sorting, and display of the event or history data.

The integrated applications that support this function include RTWorks (RTarchive, RTplayback), EPOCH2000, and OS/Comet. Database management systems include Oracle and Sybase, or may be simple flat file applications.

4.3.21.4 Geology, Star, Planet Properties

These are products defining the positions, velocities, and magnitudes of the various heavenly bodies. They can be either in flat files or managed by a database management system like Oracle or Sybase.

4.3.21.5 Plans Database

The mission planning database consists of the general plans and specific tactical plans developed to drive system operations. It may be managed using tools like Oracle or Sybase.

4.3.21.6 Product Definition Database

The product definition database contains the specifications for input data, quality, and output product characteristics to drive product generation. It may be managed using tools like Oracle or Sybase.

4.3.21.7 State and Event Definition

The monitoring databases contain the definitions for all monitoring data for the spacecraft, the ground equipment, the I&T system, and the operations center. For the spacecraft, this includes the spacecraft database that defines all telemetry format information, parameter names, associated subsystems, limits, mnemonics, data type, etc. The database may be part of an integrated product (as with EPOCH2000) or it may be a separate database application.

Section 5. Hardware Architecture

This section defines the basic hardware architecture underlying the GDS. The concept for second generation designs is to provide an overall design that not only allows for use of a wide selection of hardware but uses this variability to meet critical needs for performance and availability. It is noted here that the hardware is closely coupled to the operating system(s) that will run on the hardware and that the operating system is the potential constraint on flexibility. With this in mind, the architecture allows for implementation of two classes of components based on (at least) two operating systems. Section 5.1 identifies the hardware platform(s) and operating system(s), and other components. Section 5.2 identifies general mechanisms for integrating the distributed hardware architecture.

5.1 Hardware Components

The hardware components identified provide a generic capability as building blocks of the system. The allowed standards for interfaces are broad, requiring only that the mission system configuration be self-consistent and that the external interfaces be maintained. The components are subdivided between those that support mission activities by processing data and those whose support is data transport.

5.1.1 Computer Platform Hardware

The primary data processing components are the general computer platforms that support the software applications. These platforms will generally support any of the applications identified for Renaissance, with the mapping of applications to platforms being an operational decision. Server platforms will be used where there are important performance requirements in data storage and transport, e.g., for the software servers. Workstations and PCs will generally support any of the applications.

For the Renaissance architecture, the allowed platforms are as broad a spectrum as possible so that needs for performance can be met by as low-cost a solution as possible. The following are identified for the initial set:

- Workstation / UNIX - single and symmetric multiprocessor (SMP)
- Server / UNIX - single and SMP
- PC / MS Windows NT - single and SMP
- Workstation / MS Windows NT - single and SMP

Note that both UNIX and MS Windows NT operating systems are accepted for platforms. The expectation is that high performance systems, e.g., real-time requirements support, will generally be based in UNIX, with the Windows NT platforms used for low-cost systems where

performance is less critical. It should be noted that this allocation is not hard, because there are very high performance hardware systems available under Windows NT.

Included in the platform hardware are the directly attached peripherals that provide the platform resources. These will include standard peripherals such as monitors, disk, keyboard and local print devices. It can also include some specialized devices, such as a GPS time source, and a data acquisition interface based on a single-board computer (SBC) supporting low-level hardware testing. Embedded software on such a SBC is considered here as a very localized part of the system, encapsulated in the SBC component. Generally, these peripherals will support local platform options, while general, network-wide resources will be network-attached.

5.1.2 Network-Attached Resources

The peripheral hardware for the general computing platforms is quite extensive. Current and future systems will allow for computerized control of most system hardware. This includes the specialized hardware used to interface to payloads during test and during operations. It further includes many elements not currently computer controlled, e.g., power subsystems for both space and ground. Finally, general computer peripheral subsystems are included, specifically those that are network attached. These will include:

- Data acquisition and control: for telemetry, et al. (488, 1773,... interfaces)
- Ground station unique: RF subsystem, antenna control, digital data formatting & control, antenna
- Facility support: power subsystems, facility management
- Peripheral subsystems: RAID data server, printers

5.1.3 Standalone Hardware

Additionally, there may be independent hardware, i.e., not permanently linked to any general platform. This will normally be present in a transitory state, as for diagnosis of an observed problem. This may include:

- Test and diagnostic equipment
- Facilities equipment, e.g., power, air conditioning, security monitors
- Specialized equipment, e.g., vacuum pumps

5.1.4. Network Hardware

Network components provide the essential interconnection elements. Here they have been identified in terms of the hierarchy of networks defined, i.e., platform, LAN and WAN.

Platform to network interface:

- Network Interface Card : Ethernet, Fast Ethernet, Token Ring, Fast Token Ring, FDDI, ATM

- Modem: platform to telephone interface

Local Area Network:

- LAN transport media: co-axial cable, optical fiber, twisted-pair (UTP Grade 5), other
- LAN Hubs (Ethernet, Fast Ethernet, Token Ring, Fast Token Ring, FDDI)
- LAN switches (Ethernet, Fast Ethernet, ATM)
- LAN bridge (Ethernet, Fast Ethernet, Token Ring, Fast Token Ring, FDDI)
- LAN - Video interface

LAN to WAN interfaces:

- Protocol Gateway (may be a card in a router or switch)

Wide Area Network:

- Router (Single / Dual home)
- Switch
- Protocol Gateway
- WAN transport medium: optical fiber, microwave, copper
- Public Carrier Network Interface

Note that, generally, the mission system will not have to be involved in the details of WAN implementation, as they will be using either a public carrier or using one managed by another organization, e.g., NASA. However, for complex missions, elements of the WAN may show in constructing campus networks to integrate the system elements.

5.2 Component Interconnection

5.2.1 Platform Relationships

The computer platforms and network-attached resources are intended to be integrated by the network into a general three-tier client-server arrangement. The general arrangement for platforms is that there are some that act as servers to the other nodes on the networks, some act as human operator client platforms, and some may act as uniquely application processors. Network connected resources generally participate as servers to other platforms. All interconnections are through the network hardware. This model is illustrated in Figure 5-1.

As with the software architecture, the three-tier model is a general one, and not all platforms will conform to it. In particular, both client and server platforms overlap with application platforms, in that some algorithm processing will occur on these. For small systems, there may be an entirely peer-to-peer relationship between platforms, with common workstations or PCs to perform all automated processing.

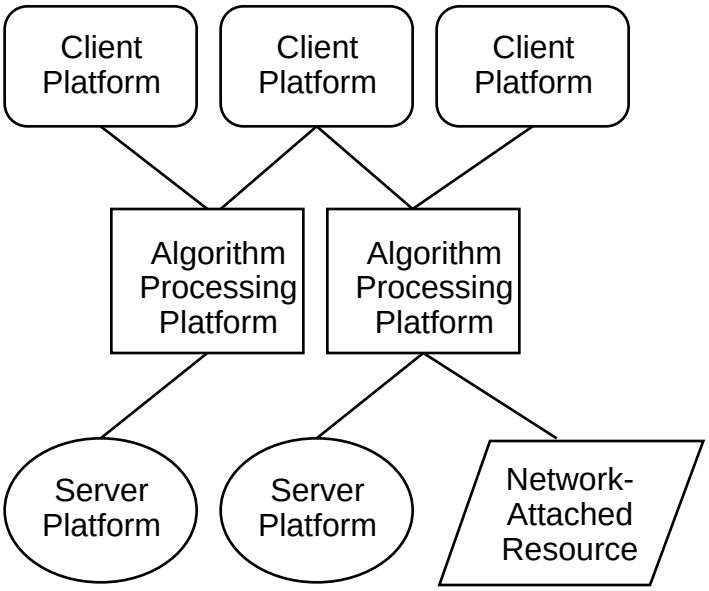

Figure 5-1. Three-tier Client-Server Hardware Model

There will also be peer-to-peer relationships between servers and between clients. Peer-to-peer sharing can occur using a number of mechanisms implemented in the network transparency layer of the software. File system sharing can be done at any level, and the file system server may not be mapped to a hardware server, though for large systems this can be important since the server platform will have specialized features to support network bandwidth for this. Also, for some applications it may be important to have direct pipelines between algorithm processing platforms to support distribution of computationally intense applications.

5.2.2 Spacecraft

The spacecraft represent one of the exceptions always present. Today's spacecraft tend to be custom hardware implementations, with embedded software that is one of the cost drivers currently. The future architecture anticipated changes to an environment in which the spacecraft is assembled from one or more computing platforms, with standard relationships, and some custom hardware for the payload and perhaps some instrumentation of the bus. The platforms would be connected by nearly standard hardware, with the caveat that it has to be able to survive the space environment. This assembly would be connected to the rest of the networked system through an extension of the mission network.

The connection of the spacecraft to the rest of the network is subject to many considerations not relevant to terrestrial networks. Because of the variations in required distance and the limitations on RF power and antenna configurations for the spacecraft, special techniques are used to achieve desired signal-to-noise ratios and effective bit error rates (BERs) for transmitted data. In particular, because of the latency inherent in long links, extended error recovery algorithms are used to minimize the need for synchronous transmission of data. Synchronization is usually achieved at the dataset level, with replay minimized through the algorithm use. For low-Earth

orbiting spacecraft, the problems are not major, and standard synchronous techniques will work. For distant spacecraft, and for those operating in domains with high RF interference, the constraints can be severe.

5.2.3 Network Connections

All platforms are interconnected through the standard mission network. The network is constructed as an assembly of one or more local area network segments, connected through an interface to a terrestrial wide area network if the system is physically distributed enough, and always via a space link to the spacecraft. The network intrconnections are illustrated in Figure 5-2. The WAN provides interfaces to the LANs, typically through a router and/or protocol gateway. Within the WAN, segments are interconnected through switches, routers and protocol gateways. Typical commercial WANs provide multiple paths between any two destinations to avoid single points of failure and to distribute loads. IP routing takes advantage of this, and routing strategies are a part of standard WAN technology.

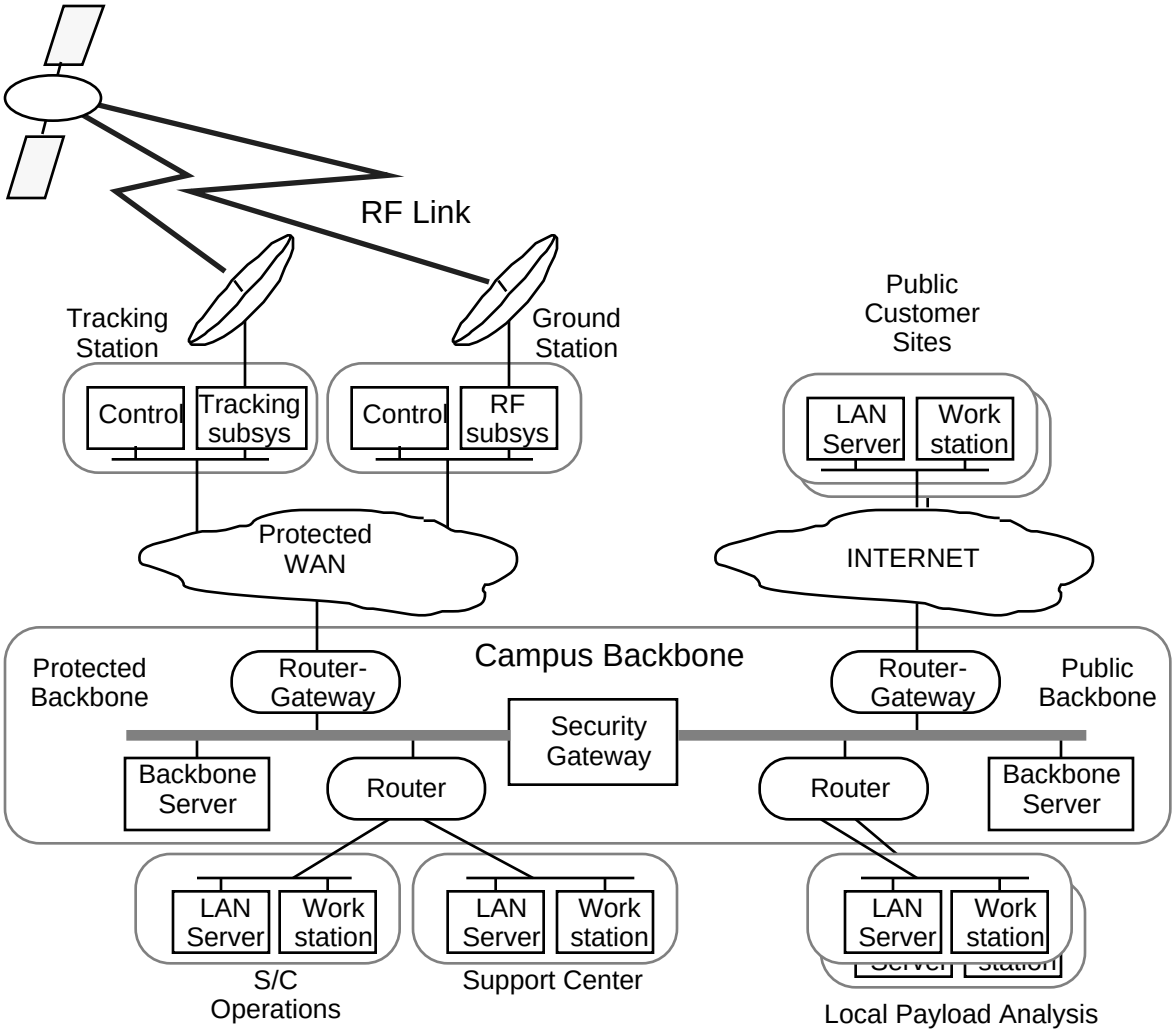

Figure 5-2. Network Connections Example

It is anticipated that almost all network hardware will be commercial. The possible exceptions are those associated with the space link. Current RF communications standards are weak at best. As a result, commercial hardware support for protocols over RF is very limited.

Abbreviations and Acronyms

3D	Three Dimensional
ADC	Analog to Digital Converter
AFS	Andrew File System
Altair MCS	Altair Mission Control System
AM-1	Ante Meridian 1 (one of the EOS satellites)
API	Application Program Interface
ASCII	American Standard Code for Information Interchange
ASIST	Advanced Spacecraft Integration and System Test System
ATM	Asynchronous Transfer Mode
BACP	Bandwidth Allocation Control Protocol
BBN	Bolt Beranek and Newman, Inc.
BER	Bit Error Rate
C&C	Command & Control
C&DH	Command and Data Handling
CCL	Comet Command Language
CCR	Configuration Change Request
CCSDS	Consultative Committee on Spacecraft Data Systems
CDE	Common Desktop Environment
CDS	Cell Directory Service
CGI	Common Gateway Interface
CM	Configuration Management
CMD	Command
COE	Center of Expertise
CORBA	Common Object Request Broker Architecture
COSE	Common Operating Software Environment
COST LESS	Code O Success Team Lifecycle Effectiveness for Strategic Success

COTS	Commercial Off The Shelf
CPU	Central Processing Unit
CRC	Cyclical Redundancy Check
CSC	Communications Services Component
CVT	Current Value Table
DAC	Digital to Analog Converter
DAD	Data Distribution component
DBMS	Database Managment System
DCE	Distributed Computing Environment
DCN	Document Change Notice
DEC	Digital Equipment Corporation
Defs	Definitions
DES	Data Encryption Standard
DFS	Distributed File System
DHDS	Digital History Data Store
DNS	Domain Name Service
DoD	Department of Defense
DRA	Data Reduction and Analysis
DSN	Deep Space Network
DTS	Distributed Time Service
EDA	Enterprise Data Access
ELV	Expendable Launch Vehicle
Email	Electronic Mail
EOS	Earth Observing System
ESA	European Space Agency
FDDI	Fiber Distributed Data Interface
FEDS	Front End Data System
FOT	Flight Operations Team
ftp	File Transfer Protocol (application)

FTP	File Transfer Protocol (protocol)
FY95	Fiscal Year 1995
FY96	Fiscal Year 1996
FYxx	Fiscal Year xx
GBytes	Gigabytes
GDS	Ground Data System
Genie	Generic Inferential Executor
GenSAA	Generic Spacecraft Analysis Assistant
GN	Ground Network
GOTS	Government Off The Shelf
GPIB	General Purpose Interface Bus
GPS	Global Positioning System
GREAS	Generic Resource, Event, and Activity Scheduler
GS	Ground Station
GSFC	Goddard Space Flight Center
GUI	Graphical User Interface
HCI	Human Computer Interface
HP	Hewlett-Packard
HPOP	High Precision Orbit Propagator
HST	Hubble Space Telescope
HTML	Hypertext Markup Language
HTTP	HyperText Transport Protocol
I&T	Integration & Test
IBM	International Business Machines, Inc.
ICCCM	Inter-Client Communications Conventions Manual
ICS	Interface & Control Systems, Inc.
ID	Identifier
IDL	Interface Definition Language
IEEE	Institute of Electrical and Electronics Engineering

IETF	Internet Engineering Task Force
IGSE	Instrument Ground Support Equipment
IMT	Intelligent Monitoring Toolkit
IO	Input Output
IP	Internet Protocol
IPv6	Internet Protocol, Version 6
IPC	Interprocess Communications
ISDN	Integrated Services Data Network
JPEG	Joint Photographic Experts Group
JPL	Jet Propulsion Laboratory
Kbps	Kilobits per second
KKI	Kisak-Kellogg Inc.
KQML	Knowledge Query and Manipulation Language
L&EO	Launch and Early Orbit
LAN	Local Area Network
LEO	Low Earth Orbit
LPC	Language Processing Component
LTIS	Loral Test & Integration Systems
MAGIC	Multimission Advanced Ground Intelligent Control
Mbps	Megabits per second
MCS	Mission Control System (Altair MCS)
MFILE	Message File
MIB	Management Information Base
MIDEX	Medium Explorer
MIME	Multipurpose Internet Mail Extensions
MLS	Multi-level Security
MMS	Manufacturing Message Specification
MO	Mission Operations
MO&DA	Mission Operations and Data Analysis

MO&DSD	Mission Operations and Data Systems Directorate
MOC	Mission Operations Center
MOCA	Mission Operations Control Architecture
MODNET	Mission Operations Directorate Network
MOPS	Mission Operations Planning System
MPEG	Moving Picture Experts Group
MS	Microsoft, Inc.
MTA	Mail Transfer Agent
MTBF	Mean Time Between Failures
MTTR	Mean Time To Restore
NASA	National Aeronautics and Space Administration
NASCOM	NASA Communications
NFS	Network File Services
NHB	NASA Handbook
NIC	Network Interface Card
NIST	National Institute for Standards and Technology
NMS	Network Management System
NT	New Technology
OAC	Orbit Analysis Component
OB	Onboard
ODBC	Open Database Connectivity
OODBMS	Object Oriented Database Management System
ORB	Object Request Broker
OS	Operating System or Open System
OSE	Open System Environment
OSF	Open Systems Foundation
OSI	Open Systems Interconnect
OTS	Off The Shelf
Pacor II	Packet Processor II

PC	Personal Computer
PCI	Peripheral Component Interconnect (bus)
PDC	Programmable Device Controller
PDP	Production Data Processing
PFS	Platform File Services
PGP	Pretty Good Privacy
PI	Principal Investigator
pixmap	Pixel Map
PODS	Precision Orbit Determination System
PPS	Packet Processing System
PTCW	Primary Test Conductor Workstation
RAID	Redundant Array of Inexpensive Disks
RDBMS	Relational Database Management System
REM	Remote Equipment Monitor
Renaissance	Reusable Network Architecture for Interoperable Space Science, Analysis, Navigation, and Control Environment
RF	Radio Frequency
RFC	Request For Comment
RISC	Reduced Instruction Set Computer
RLC	Recording & Logging Component
RMON	Remote Monitoring Protocol
RMP	Reliable Multicast Protocol
ROM	Read Only Memory
RPC	Remote Procedure Call
S/C	Spacecraft
SA	Single Access
SAC	System Administration Component
SBC	Single Board Computer
SCL	Spacecraft Command Language

SCPS	Space Communication Protocol Standards
SFDU	Standard Formatted Data Unit
SMEX	Small Explorer
SMEX-Lite	Small Explorer - (Lower Cost Version)
SMP	Symmetrical Multiprocessor
SMTP	Simple Mail Transfer Protocol
SN	Space Network
SNMP	Simple Network Management Protocol
SNMPc	Simple Network Management Protocol - Personal Computer
SNMPv1	Simple Network Management Protocol Version 1
SNMPv2	Simple Network Management Protocol Version 2
SONET	Synchronous Optical Network
Specs	Specifications
SQL	Standard Query Language
STI	Software Technology, Inc.
STK	Satellite Toolkit
STOL	System Test and Operations Language
SuperMOCA	Space Project Mission Operation Control Architecture
SYM	Symbol Processing component
TBD	To Be Determined
TBR	To Be Resolved
TBS	To Be Supplied
Tcl	Tool Control Language
TCP	Transmission Control Protocol
TCW	Test Conductor Workstation
TDRS	Tracking and Data Relay Satellite
TLI	Transport Level Interface
TLM	Telemetry
TPCE	Telemetry Processing Control Environment

TPOCC	Transportable Payload Operations Control Center
TSDS	Decommutated Sequential Telemetry Stream Server
TSTOL	TPOCC System Test and Operations Language
UDP	User Datagram Protocol
UID	User Interface Description
UIL	User Interface Language
US	United States
UTC	Coordinated Universal Time
UTP	Unshielded Twisted Pair
VC	Virtual Channel
VCID	Virtual Channel Identifier
VCDU	Virtual Channel Data Unit
VME	Versa Module European
VUE	Visual User Environment
WAN	Wide Area Network
WWW	World Wide Web
X11R5	X Window System Release 5
XTE	X-Ray Timing Explorer

Glossary

Application	A piece of software that provides a primary mission support capability, e.g., spacecraft command and control.
Cell	The highest level of mission system component, implemented using the standard local area architecture and technology.
Center of Expertise	A system support entity that provides the mission with expert support for development or operations beyond that which is normally available among the mission operations team. COEs may be used for nominal support or may be called upon for contingency situations.
Client	A software process that exchanges information with a server. In the Renaissance second generation architecture, clients include applications software as well as other servers.
Client-Server	A form of distributed processing in which multiple clients interact with a server process controlling an information resource.
Customer	The person, organization, or group which provides the source of funds for developing of the mission system. Indirectly, the organization or community which is using the space system to address a research problem or provide a commercial service. The funding and using customer may be the same organization.
Dataset	A collection of information that is generated for use at an arbitrarily later time. Data delivered as datasets include software images, data logs, and customer products. Subsets of the entire collection are typically extracted for use.
Dataset Server	A server that handles the distribution of complete or partial datasets from producers to consumers.
Dedicated	Describes a resource that is developed and operated in support of a single mission. A dedicated resource is totally managed by the mission.
Domain	A sphere of activity within the system, within which common services are provided.
Extended Operations	The phase of the mission life cycle, after the mission's primary goals have been met, in which secondary goals are pursued. Extended operations usually begin after some number of months or years after launch.
External	Anything that is not provided as part of a standard cell. External interfaces typically provide access to shared and contingency resources.

	Interface types may include data interfaces, schedule interfaces, operational messages, and management interfaces.
Front End	System that performs protocol conversion between terrestrial communications and space-to-ground communications for telemetry (return link) and commands (forward link). Currently, front ends are comprised of special hardware and software.
Ground Data System	A portion of the mission system, located on the ground, that is responsible for capture and processing of data returned by the spacecraft and for the preparation and uplink of commands and data to the spacecraft.
Ground Station	A system that provides data transport and protocol conversion between terrestrial communications networks and the RF communications necessary to communicate with spacecraft after launch. Ground stations are typically composed of one or more antennae and RF equipment. The definition is expanded to include the ground elements of systems such as the Space Network which use relay satellites to communicate with spacecraft in orbit. Ground stations may include tracking support as a supplemental function.
Integration & Test	The phase of the mission life cycle prior to launch during which the spacecraft components are integrated and checked out, followed by mission system establishment and testing.
Internet	A world-wide, public network used for interchange of data and information.
Internet Protocol	A suite of communications protocols and applications developed for transmission of information across the Internet. Also widely used for transmission on local area networks and private wide area networks. More specifically, the protocol which provides the network layer in the OSI model, covering packet addressing and routing.
Local Area Network	A network of nodes (addresses) within a single domain of lowest level Internet Protocol addresses, usually implemented with technology appropriate to small (~100 m) networks.
Message	A packet of data containing information that must be acted upon promptly. Data delivered as messages include real-time telemetry and commands and operations event data
Message Server	A server that handles the delivery of distinct messages from producers to consumers.
Mission	A coordinated effort to develop and operate a spacecraft (or group of related spacecraft) in support of an overall scientific goal.

Mission Operations	All activities supporting accomplishment of mission goals, including provision of customer data, mission system resource management, and system evolution.
Mission System	The collection of dedicated and shared resources needed to carry out the goals of a mission. The mission system consists of both ground and space resources.
Mission System Management	The operation and control of dedicated resources and scheduling and monitoring of shared resources used to meet the mission's goals.
Multi-Mission	A coordinated effort to develop and/or operate two or more missions using the same resources.
Off-the-Shelf	Hardware or software that can be integrated into a system "as is" or with only configuration modifications. Includes commercially available (Commercial Off-the-Shelf or COTS) and government-developed (Government Off-the-Shelf or GOTS).
Operations Phase	The phase of the mission life cycle, after launch, during which the mission's primary goals are being pursued.
Operations Center	A facility within the mission system that is responsible for some portion of the mission operations, normally implemented as a single cell.
Payload Analysis	Data processing and analysis of payload generated data. Analysis includes technical analysis of data to pursue primary customer goals as well as engineering analysis of data to monitor instrument health and safety.
Peer-To-Peer	A form of distributed processing in which applications communicate directly with each other as equal partners.
Publish/Subscribe	A client-server interaction method in which consumers receive requested types of messages or data in continuing streams, receiving new messages or data updates as the server receives them from producers, or on a periodic basis.
Request/Response	A client-server interaction method in which message consumers determine what messages or data they want by making an individual specific request to the server for the data.
Server	A software process that manages an information resource, e.g., an information repository or the human-computer interface.
Shared	A resource that is developed and operated in support of multiple missions. Examples include multi-mission operations centers, institutional ground stations, and the Internet. Shared resources are managed by an entity external to the mission.

Subdomain	A group of tightly-coupled applications within a specific cell. Subdomains are characterized by a specific, shared data interface which may vary from the design standard client-server model.
Wide Area Network	A network that is used for cell-to-cell communication and cell-to-external communication.

Bibliography

Application Portability Profile, OSE/1 Version 2.0, NIST Special Publication 500-210, 1993

The Essential Distributed Objects Survival Guide, Orfali, Harkey and Edwards, John Wiley & Sons, Inc., 1996, ISBN 0471-12993-3

Guide to Writing DCE Applications, Shirley, Hu and Magid, O'Reilly & Associates, Inc., 1994, ISBN 1-56592-045-7